SENSITIVITY ANALYSIS FOR ONLINE MANAGEMENT OF PROCESSOR POWER AND PERFORMANCE

A Thesis
Presented to
The Academic Faculty

by

Nawaf I. Almoosa

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering

Georgia Institute of Technology
May 2014

SENSITIVITY ANALYSIS FOR ONLINE MANAGEMENT OF PROCESSOR POWER AND PERFORMANCE

Approved by:

Dr. Sudhakar Yalamanchili, Advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Yorai Wardi, Co-Advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Karsten Schwan
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Magnus Egerstedt
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Saibal Mukhopadhyay
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Santosh Pande
College of Computing
Georgia Institute of Technology

Date Approved: January 3, 2014

To Mahra.

ACKNOWLEDGEMENTS

I would like to extend my sincere gratitude to my advisors, Dr. Sudhakar Yalamanchili and Dr. Yorai Wardi, for their motivation, guidance and unwavering support throughout my life as a Georgia Tech graduate student. My growth as a researcher was greatly shaped by their work ethic and emphasis on research integrity, and what I have learned from them is way broader to be restricted to the academic domain. I am really thankful to Dr. Yalamanchili and his dear wife Mrs. Padma Yalamanchili for their support during the turbulent days of early parenthood, and to Dr. Wardi for instilling the love of nature and the outdoors, and introducing me to some of my favorite places on this continent and possibly the globe, the national parks at the state of Utah.

Dr. Santosh Pande, Dr. Saibal Mukhopadhyay, Dr. Magnus Egerstedt, and Dr. Karsten Schwan formed my defense committee. I would like to thank them for the insightful feedback that helped strengthen my contributions. I have immensely the Data Compression and Analysis course taught by Dr. Biing-Hwang (Fred) Juang, which kickstarted a collaboration with his then Ph.D student Soo Hyun Bae. I learned from and deeply enjoyed this experience, and thanks are due to Dr. Juang and Dr. Bae.

This dissertation would have not been possible without the support from the folks at Khalifa University for Science, Technology and Research (KUSTAR). I am deeply grateful for the support of Dr. Khalid Mubarak, Dr. Muhammad Al-Mualla and Dr. Arif Al-Hammadi.

I would also like to thank my colleagues at the Computer Architecture and Systems Laboratory (CASL) and Georgia Tech in general. The support and friendship of

Tushar Kumar and Dr. Jeff Young was instrumental in seeing this work through. Thanks are also due to Mitchelle Rasquinha, Subramanian Ramaswamy, William Song, Dhruv Choudhary, Dr. Andrew Kerr, and Dr. Greg Diamos.

I would like also to thank my extended Atlanta family for their friendship and constant support, especially John Cole, John Fogleman, Michael Packard, Julia Schneider, Tyler Brown, Kelly Mccormick, Iman Msallak, Chasity West, and Kamal and Zuni and the kids.

Finally, I would like to thank my family for their support and understanding during the long years of graduate school. To my father Ibrahim, my mother Maryam, and my wife Aaesha: I owe you a debt of gratitude that will never be fulfilled. To my daughter Mahra: the sky is the limit.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Processor designers adopted manycore architectures to curtail the rising power-consumption levels and maintain performance scaling via parallelism. Despite its success, the manycore paradigm has introduced unprecedented challenges to the design and operation of processors. The diverse applications that execute simultaneously on manycore platforms cause high runtime power and performance variations. Power variations impact the reliability of chips, as well the cost of their cooling and power-delivery systems. Moreover, the variability of performance has an impact on the energy efficiency and performance returns of manycore processors. The aforementioned challenges have highlighted the need for runtime power and performance management of manycore processors. However, the design of management algorithms is challenging since power and performance are strongly dependent on the workload, which cannot be determined *apriori* and exhibits wide and rapid runtime variations.

This dissertation seeks to show that sensitivity analysis provides runtime information about the time-varying power and performance behaviors that enables the design of adaptive management algorithms for manycore processors. Towards this goal, the dissertation contributes adaptive algorithms that rely on runtime sensitivity (derivative) estimation to solve the problems of controlling the power and performance of processor cores, maximizing the performance of manycore processors under a fixed power budget, and optimizing the energy consumption of cache memories.

The first contribution is concerned with controlling the power of processor cores, which is an essential component of controlling the power of manycore processors and maximizing their performance. The design of core-power controllers that guarantee

stability and rapid settling is challenging since power is a function of the time-varying workload. The dissertation proposes an integral controller that tracks desired power levels by adjusting core frequency settings. The derivative of the time-varying plant (frequency-power functional relation) is estimated online and is used to adaptively set the controller gain. The proposed adaptive controller is shown formally and via detailed simulation to achieve rapid and robust tracking under diverse workload conditions. In contrast, simulation results show that plant variations can degrade the settling time of fixed-gain controllers designed using offline analysis.

The next contribution is concerned with the problem of regulating application throughputs via adjustment of core frequency settings. Throughput targets are set to ensure quality of service and improve energy efficiency. The design of throughput regulators is however challenging due to the wide range of application behaviors, and the throughput fluctuations introduced by the memory hierarchy and speculative execution employed in out-of-order processing. The dissertation proposes an integral-control algorithm for throughput regulation, where the gain is adaptively set using the sample derivative of the frequency-throughput functional relation that is found via infinitesimal perturbation analysis (IPA). Simulation results show that the proposed algorithm can precisely regulate the throughput of out-of-order processors under a wide range of workload variations.

Next, the dissertation proposes a solution to the problem of chip power control, which is concerned with maximizing the performance of manycore chips while ensuring the power envelope stays below the chip power budget. Solving the chip power control problem has central implications on the design of the processor cooling and power delivery systems, and maximizing the performance gains of the manycore processing paradigm. The proposed solution methodology decomposes chip power control into a *master* problem, which is concerned with calculating a performance-maximizing partition of the chip power budget between cores, and *regulation* subproblems, which are

concerned with tracking the fractions of the power budget (power setpoints) assigned to each core. The regulation subproblems are solved using the core-power regulators described earlier. The master problem is solved using an iterative constrained-optimization scheme based on the method of gradient projection that calculates the power setpoints using the sample power-throughput derivative calculated at each core. The power setpoints calculated by the master algorithm satisfy the chip power budget constraint, as well the the core-level finite DVFS settings. Simulation results, performed using a detailed multicore processor simulator and industry-standard benchmarks, show that the proposed solution controls the power envelope more precisely and yields higher performance than state of the art.

Finally, to optimize the energy consumption of cache memories, we propose an iterative optimization algorithm based on the method of gradient descent. The proposed algorithm is an adaptive version of the cache decay technique , which switches off cache lines predicted to be unused to save their leakage energy, but may result in energy overheads in case of mispredictions. The proposed algorithm adaptively sets the cache-decay parameters to balance the tradeoff between cache leakage-energy savings and the energy overheads of induced misses, thereby optimizing cache energy under variable access patterns.

The application of the proposed adaptive algorithms within a manycore processor is illustrated in Fig. 1, and a summary of the role of sensitivity analysis across the tackled problems is provided in Fig. 2. Technology trends point to growing behavioral variability in manycore processors, due to the increased levels of core integration and the adoption of heterogenous architectures that combine few complex out-of-order cores with numerous simple in-order cores. The increased behavioral variability presents strong incentives for adopting sensitivity-based algorithms that achieve adaptive, low-complexity, and robust management of manycore processors.

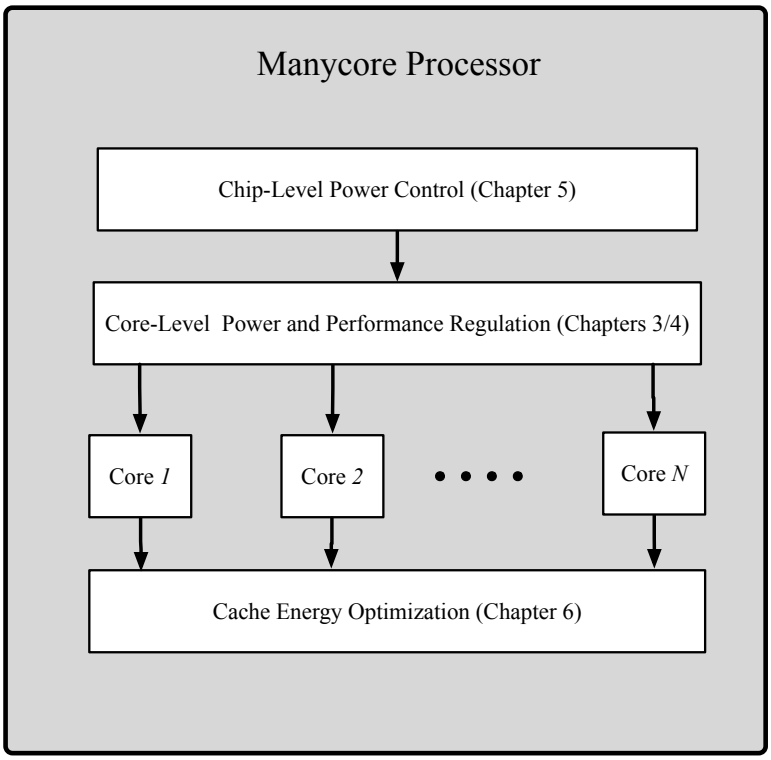


Figure 1: Application of the proposed algorithms in a manycore processor

Problem	Unknown	Solution
Core Power Control	Frequency-Power Relationship	Power Derivative
Throughput Regulation	Frequency-Throughput Relationship	Sample Throughput Derivative
Chip Power Contol	Feasible Power Range	interpolation via power derivative
	Peak Core Throughput	interpolation via throughput derivative
	Performance Objective Function	Power-Performance Derivative
Cache-Energy Optimization	Cache-Energy Function	Cache-Energy Sensitivity

Figure 2: Problem unknowns addressed via sensitivity analysis

CHAPTER I

INTRODUCTION

Microprocessors have historically enjoyed exponential performance growth due to device-geometry scaling (Moore's Law), which allowed fabricating smaller and faster transistors [1], and Dennard scaling [2], which allowed scaling supply voltages to maintain affordable chip power envelopes. The diminishing voltage-scaling margins in the past decade, coupled with the static power-rise [3], have drastically elevated power consumption, and risked operating in temperature regimes beyond the capability of existing cooling solutions [4]. This so called *Power Wall* has disrupted the trend of performance growth via frequency scaling, and presented serious challenges to the economic model of the semiconductor industry, where the great costs of developing new technology nodes are justified by performance returns [5].

Processor designers adapted to these power challenges by changing the microprocessor blueprint from a *single* high-power, high-speed core, to *multiple* reduced-power, reduced-speed cores [6]. This *multicore* shift aims at curtailing power growth and realizing the performance utility of transistor scaling via parallelism, instead of processor speeds. While the shift has been so far successful in averting the negative outcomes of the power wall, the performance of computing systems remains largely constrained by power consumption [7].

In the embedded computing domain, which is characterized by an increasing demand for performance and modest improvements in battery capacities, minimizing power consumption is a central design goal to improve the energy efficiency of platforms [8]. A pressing need for energy efficiency is also present in large-scale data centers, where the escalating energy costs are bringing to bear heavy financial costs

and negative environmental impacts [9]. Moreover, the number of housed servers, which represents the performance return on the high setup and operating costs of data centers, is limited by the capacity of the power distribution and cooling systems [10, 11, 12, 13]. In fact, the performance potential of multicore processors *in general* will be constrained by power consumption, since a fraction of the cores may need to be turned off to ensure the power envelope is below the chip power budget [7, 14].

The manycore shift has also introduced unique power and temperature challenges to the design and operation of microprocessors. Due to the diverse applications that execute simultaneously, manycore processors generate increasingly-variable power envelopes that are prone to exceeding the chip power budget and generating heat beyond the capacity of the processor cooling system. Heat has a detrimental effect on chip reliability, since it is linked to many mechanisms behind transient and permanent processor faults such as timing failures and accelerated wear [15], and catastrophic chip failures due to thermal migration [3]. Moreover, the distributed nature of manycore execution generates spatially-nonuniform thermal fields, which reduces the effective capacity of the cooling system and further undermines chip reliability due to mechanical stress [16, 3]. Heat tolerance can be potentially improved with technologies such as liquid cooling or refrigeration [16, 4]. Widespread adoption of these technologies is however unlikely in the short term due to market considerations such as cost and form factor [16].

The increased variability in the power envelope has emphasized the need for *runtime* techniques that manage power and performance while the processor is executing [17]. Runtime management can be achieved with several processor control variables, the most effective of which is dynamic voltage-frequency scaling (DVFS) [18, 19, 20] , since reducing the frequency and voltage of a processor in tandem leads to cubic power reductions [4]. There are several challenges underlying the design of DVFS-based management algorithms. First, the implementation environment (hardware or

firmware) has limited computational resources and faces fast workload variations. The low computational complexity and running time of the algorithmic solution are therefore important, as well as its scalability given the projected increase in the number of cores on chip [21, 14]. Second, an integral step in management-algorithm design is characterizing the relationship between the control variables and the outputs of the processing system, namely power and performance. System characterization is a fundamental challenge since the state of the processor varies rapidly and widely due to the diverse program behaviors encountered at runtime. Within the state of the art, system characterization was carried out either via *offline analysis* [17, 22, 23, 14], or by constructing an online model using runtime observations [24]. Offline analysis tends to undermine the adaptability of management algorithms, since the mismatch between online and offline workload conditions is highly likely [25, 26]. On the other hand, empirical online estimation increases the complexity of the management scheme, especially in manycore processors where the problem size (number of cores) is projected to increase [14].

This dissertation contributes adaptive management algorithms for regulating the power and performance of processor cores, maximizing the performance of manycore processors under a fixed power budget, and optimizing the energy of cache memories. The central feature of the proposed algorithms is the use of sensitivity analysis (derivative estimation) to provide runtime power and performance information to adapt to the rapid workload variations. Thus, this dissertation seeks to show that sensitivity analysis enables the design of effective and scalable power and performance management algorithms for manycore processors.

To regulate the power of processor cores, we propose a DVFS-based integral controller whose gain is adaptively set using the derivative of the frequency-power functional relation found via sensitivity analysis. Formal analysis and simulation results show the rapid settling time and robustness of the proposed core-power regulator.

Next, we propose a DVFS-based algorithm for application-throughput regulation that improves the energy efficiency and performance predictability of execution. The proposed algorithm is an integral controller with an adaptive gain set using the sample derivative of the frequency-throughput relation calculated via infinitesimal perturbation analysis (IPA). Simulation results show that the proposed algorithm tracks the targeted throughput under rapid workload variations.

Next, the dissertation proposes a solution to the problem of chip power control, which is concerned with maximizing the performance of manycore chips while ensuring the power envelope stays below the chip power budget. Solving the chip power control problem has central implications on the design of the processor cooling and power delivery systems, and maximizing the performance gains of manycore processors. The proposed solution methodology decomposes chip power control into a *master problem*, which is concerned with calculating a performance-maximizing partition of the chip power budget between cores, and *regulation subproblems*, which are concerned with tracking the fractions of the power budget (power setpoints) assigned to each core. The regulation subproblems are solved using the core-power regulators described earlier. For the master problem, we propose an iterative constrained-optimization scheme based on the method of gradient projection that calculates the power setpoints using the sample power-throughput derivative at each core. The power setpoints calculated by the master algorithm satisfy the chip power budget constraint, as well the the finite core-level DVFS settings. Simulation results, performed using a detailed multicore processor simulator and industry-standard benchmarks, show that the proposed chip-power control solution controls the power envelope more precisely and yields a higher performance margin than state of the art,

Finally, to optimize the energy consumption of cache memories, we propose an iterative optimization algorithm based on the method of gradient descent. The proposed algorithm is an adaptive version of the cache decay technique [27], which switches

off cache lines predicted to be unused to save their leakage energy, but may result in energy overheads in case of mispredictions. The proposed algorithm adaptively sets the cache decay parameters to maximize energy savings under a variable workload.

The remainder of this chapter reviews the literature of power and performance management problems. Chapters 3 and 4 discuss the sensitivity analysis and regulation algorithms for power and performance, respectively. Next, the proposed chip power control solution is presented in Chapter 5. The adaptive cache-decay algorithm is then presented in Chapter 6, followed by the conclusions and future research avenues.

1.1 Related Work

DVFS-based proposals in the literature tackle a variety of power and performance management problems. A class of proposals was concerned with regulating application throughputs to improve the performance predictability and the energy efficiency of execution [8, 28, 29, 30, 31, 32]. Lu *et. al.* used synthetic workloads to demonstrate the efficacy of DVFS in stabilizing the throughput and improving the energy efficiency of multimedia workloads [8]. Wu *et. al.* [28] and Juang *et. al.* [29] proposed schemes for controlling the queuing occupancy of processors by adjusting core DVFS settings. Throughput regulation can be potentially achieved using these schemes, since the queuing occupancy maps to throughput. However, it is challenging to determine the mapping between queuing occupancy and throughput since it is application dependent.

Suh *et. al.* proposed adjusting the DVFS settings to regulate throughput using a proportional-integral-derivative (PID) controller [31]. The controller gains are calculated offline using a task training set. The authors report reasonable tracking provided the runtime workload does not deviate significantly from training set, limiting application to known workloads. Herbert *et. al.* [33] proposed a heuristic that searches the DVFS space of a multicore processor for a combination that maximizes energy efficiency. In contrast to the aforementioned works, the performance-regulation algorithm reported in Chapter 4 is concerned with tracking throughput targets specified externally; *e.g.* by the operating system (OS), via an integral controller whose gain is calculated *adaptively* using the sample-throughput derivative.

DVFS is also the basis of several proposals for *chip power control*, which is a constrained optimization problem concerned with maximizing the performance of many-core processors while ensuring that the power envelope is kept below the chip power budget [17, 34, 35, 21, 22, 23, 24, 36, 14]. Solving the chip power control problem

has central implications on the physical design of processors and their power delivery and cooling systems, and on maximizing the performance gains of the multicore paradigm [14]. Intel Montecito; a dual-core Itanium processor, was equipped with a control system comprising power and temperature sensors and a dedicated microcontroller, which successfully controlled power and temperature using chip-wide DVFS [37]. Subsequently, most of the commercial processors came equipped with per-core DVFS settings, which improved the control accuracy. Heuristics were the basis of several chip-power control proposals in the literature [17, 34, 35, 21]. The heuristic approaches included trial-and-error adjustment of core DVFS settings [17, 21], and exhaustive search of the per-core DVFS space using offline-generated power and performance predictive models [17]. Later works [24, 14] have shown the drawbacks of heuristic-based approaches, which include prolonged algorithm running times, imprecise control, and limited performance. Feedback-control algorithms were adopted by several proposals to increase the robustness to model estimation errors and provide theoretical guarantees on algorithm performance. Mishra *et. al.* [22, 23], proposed a power-budgeting scheme where the chip power budget is partitioned between voltage islands to maximize chip performance. The power setpoints assigned to each voltage island are tracked by adjusting the local DVFS settings using PID control. Stability of the local PID controllers is not guaranteed since they are designed using average offline analysis. Moreover, since it does account for the limited DVFS settings available at each voltage-island, the scheme may calculate infeasible power partitions.

Wang *et. al.* [24, 36] proposed a centralized scheme based on model-predictive control (MPC) [24]. The relationship between core DVFS settings and chip power was modeled as a linear memoryless dynamical system, whose parameters were estimated empirically at runtime. The authors demonstrated successful power and temperature control on a quad-core Intel Xeon processor. However, the scheme yielded high computational complexity and algorithm running time due to its centralized nature and

the online model estimator which requires matrix inversion [22, 23, 14].

Ma *et. al.* [14], proposed a power control scheme for many-core processors running single and multi-threaded workloads. The scheme uses an integral controller that adjusts the chip *frequency quota*, defined as the aggregate frequency of all the cores, to ensure the chip power envelope tracks the budget. The gain of the integral controller is calculated offline to guarantee stability by assuming worst-case plant conditions. The chip frequency quota calculated by the integral controller is then partitioned between the cores to maximize *fair* performance. The frequency-quota partitioning algorithm does not account for the limited range of DVFS settings available at each core. Thus, it may calculate infeasible DVFS settings that negatively impact chip power tracking and performance. Moreover, the reliance of the scheme on offline analysis, both in the design of the power controller and in estimating application performance necessary for fair performance maximization, makes the scheme less adaptive to the rapid variations in power and performance behaviors at runtime. In contrast, the chip power control scheme proposed by this dissertation achieves adaptation by characterizing the power and performance behaviors online using derivative estimation, it accounts for the constraints imposed by limited core DVFS settings. The simulation results, using a detailed microprocessor simulator and industry standard benchmarks, presented in chapter 5 show that the proposed power-control scheme can achieve higher power tracking accuracy and fair chip performance than the state of the art scheme proposed in [14].

The literature also includes works that used feedback control in conjunction with DVFS to control the power of high-density servers [10], enclosures [12], and large-scale data centers [11, 13]. The works described in this dissertation are concerned with power and performance management of processors. Extending sensitivity analysis for other compute settings is a potential avenue for future research.

Improving the energy efficiency of cache memories using adaptive mechanisms has

been the goal of several proposals [27, 38, 39], which attempt to minimize unnecessary leakage-energy costs during cache operation. Cache decay [27] is an energy-efficiency enhancement technique that minimizes the residency times of *dead* cache lines, which consume leakage energy in the interval between the last hit and line eviction. It predicts a cache line to be dead if it remains idle for a duration longer than a threshold termed the *decay interval*, and subsequently turns the line off using a Gated-Vdd mechanism. However, non-ideal prediction of dead lines induces extra cache misses that lead to energy and performance overheads. Moreover, the calculation of the cache-decay value that balances the tradeoff between leakage-energy savings and induced-miss energy losses requires an adaptive scheme since it varies with the workload. [38] proposed an adaptive cache-decay scheme that controlled the induced miss rate caused by cache decay using an ad-hoc algorithm. Subsequently, [39] proposed a proportional-integral (PI) controller for tracking the induced miss rate. Instead of setting a reference miss rate, the work described in Chapter 6 proposes an adaptive mechanism that used the gradient-descent algorithm to directly set the cache-decay interval to maximize cache energy savings, while minimizing the energy impact of induced cache misses.

CHAPTER II

MANYCORE PROCESSORS OVERVIEW

Manycore architectures were adopted by processor designers after single-core performance growth, which was sustained by increasing clock frequencies and power-inefficient instruction-level parallelism (ILP) extraction techniques, became practically infeasible due to excessive power dissipation [4]. Manycore processors aim at exploiting technology scaling to deliver power-efficient performance growth via parallel execution of single and multiple-threaded applications on an increasing number of cores [40]. A high-level view of manycore processors is shown in Fig. 3, which are comprised of multiple cores and cache memories that communicate via an on-chip interconnect. The manycore processor is connected via a bidirectional bus interface to dynamic random access memory (DRAM).

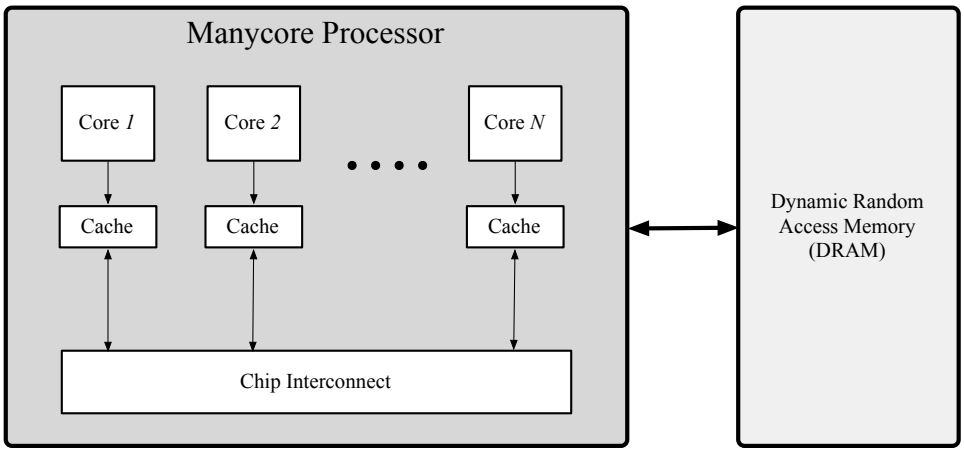


Figure 3: Block diagram of a manycore processor

Cores may execute in *out-of-order* or *in-order* fashion. In-order cores, shown in Fig. 4, employ a simple pipeline where the instructions are fetched, executed, and retired in the same order. Their simple microarchitecture yields reduced area and power consumption compared to out-of-order processors, but results in lower number

of executed instructions per cycle (IPC).

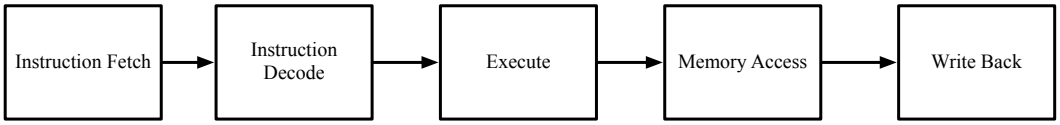


Figure 4: In-order core pipeline

Out-of-order cores, whose high-level view is shown on Fig. 5 relax the order of execution to exploit instruction-level parallelism (ILP), and employ aggressive pipelining and speculative execution to maximize the average number of instructions executed per cycle. Out-of-order cores yield higher performance compared to in-order cores, especially when executing serial code segments, at the cost of higher area and power consumption.

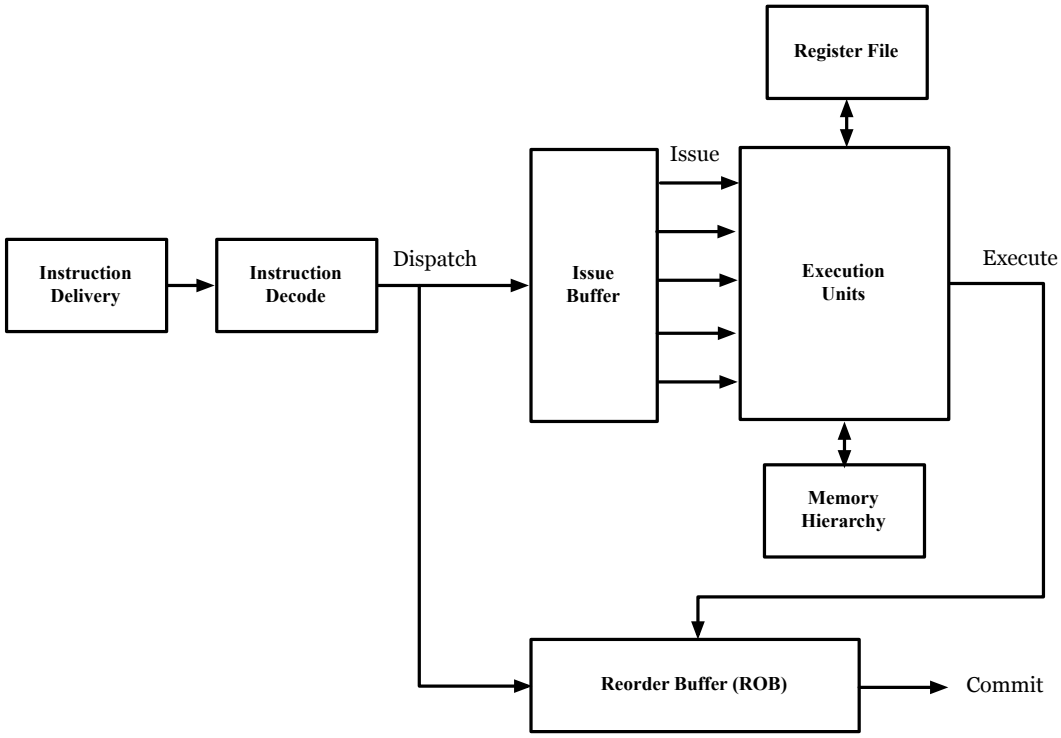


Figure 5: High-level of an out-of-order core

While several processor products are configured in a homogenous fashion, where

all the cores share the same microarchitecture, there are strong incentives for heterogenous (asymmetric) manycore processors that provide a mix of core microarchitectures that deliver comparable performance to homogenous manycore processors at lower power densities [41], which reduces the cost of the cooling system and improves its efficiency [16].

While the manycore-processing paradigm has reduced the rate of chip power growth, power consumption remains a major constraint on the design and operation of processors. The major components of processor power consumption are due to switching, leakage, and short-circuit losses [4]. Switching-loss (dynamic) power is dissipated due to the switching of logic gates, and is a function of the capacitance and activity factor of the processor, as well as its supply voltage and clock frequency. Dynamic power was the dominant power component in pre-180 nm technologies. Since then, technology scaling has been accompanied by an increased leakage-power component due to the reduction of threshold voltages [4]. Despite the power-dissipation growth of other system-level components such as DRAM, which consumes up to 40% of server-level power, the power consumption of computing system is still largely dominated by processors [42].

The increased processor power consumption, both in its dynamic and leakage components, resulted in elevated chip-temperature levels that present challenges to the design of the chip cooling system [16, 3]. The capacity of the cooling system translates to a thermal-design power (TDP) that must not be exceeded by the chip at runtime. The TDP is set to reflect the power consumption of realistic execution scenarios [16], which yields significant cost savings compared to over-provisioned cooling systems that target the rarely-occurring peak processor power consumption. However, the chip power envelope is unpredictable at runtime and may exceed the prescribed TDP, which negatively impacts chip reliability. Processors are therefore equipped with power control variables, the most effective of which has been core

dynamic voltage-frequency scaling (DVFS) [18, 19, 20, 4], to ensure the chip power envelope is inline with the cooling system specifications.

To maximize the performance returns of manycore scaling, runtime chip power control must be carried out with minimal performance impact. Quantifying the performance of manycore processors is difficult since it involves balancing single-program performance with overall system throughput [43]. Moreover, it must be capture other goals such as fairness, which ensures that all programs coexecuting on the manycore platform and share its resources experience equal benefit relative to when they are executing individually [43]. An initial manycore performance performance was *IPC throughput*, which is defined as the sum of the IPCs of the executing programs. The drawback of the IPC-throughput metric was that it did not capture an notion of fairness, and therefore can be maximized by favoring applications with inherent high-IPC. Recently, the harmonic mean of IPC speedups was proposed by [44] and adopted by several works [45, 14] as a performance metric that captures a better notion of fairness. However, the quantification of performance and fairness in a manycore setting remains an open research problem [43].

CHAPTER III

CORE POWER REGULATION

3.1 Overview

The power constraints and the distributed nature of computing systems have highlighted the need for precise local power control. In manycore processors, power control at the core level is an essential component of budgeting schemes that partition the chip power budget between cores to maximize performance [22, 23]. In the data-center domain, power control at the blade (server) level is critical to reducing the costs of cooling and enabling power shifting between servers to maximize data-center performance [46, 11, 13].

Several power control proposals relied on adjusting DVFS settings using offline model-driven feedback control algorithms [46, 11, 13, 24, 22, 23, 14]. Offline model construction is illustrated in Fig. 6, and it involves running a representative set of programs on the compute platforms to generate a model of the relationship between DVFS settings and power (plant). The offline plant model is then used to set the parameters of the power control algorithm. Proportional [46], and integral [11] control laws were proposed to control the power of blade servers based on an offline linear plant model. A similar plant modeling approach was adopted in [22, 23] to design a proportional-integral-derivative (PID) algorithm to control the power of voltage-islands in chip multiprocessors. Given the wide range of program characteristics, the plant experiences high variations at runtime, which may negatively impact the tracking of offline-designed power controllers.

This chapter introduces an adaptive law for controlling the power of processor cores using DVFS. The plant is assumed to be convex, and is controlled using an

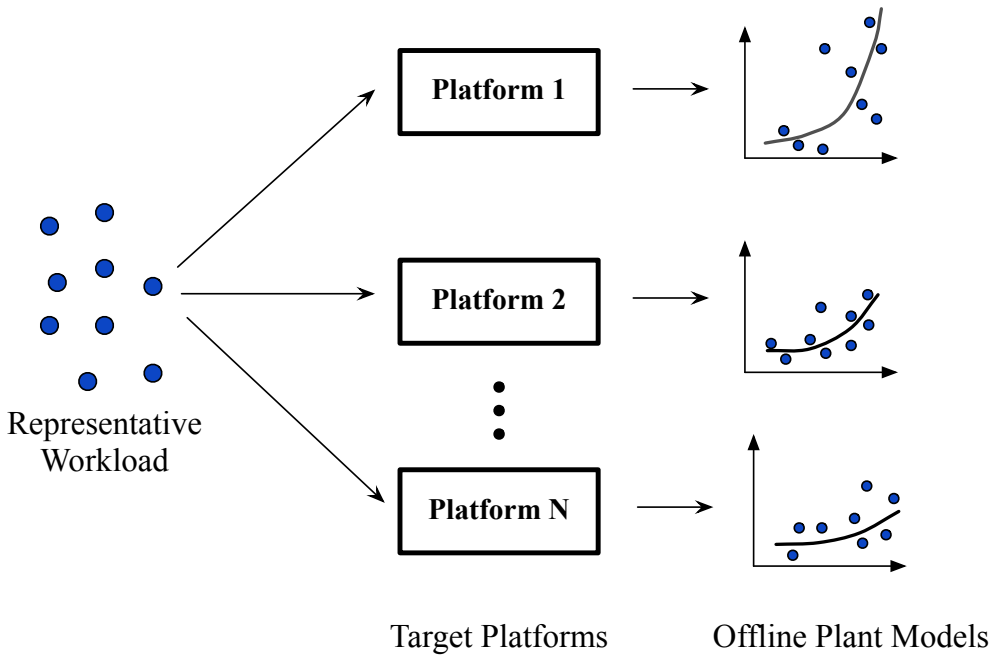


Figure 6: Offline plant characterization

integral controller whose gain is calculated online via estimation of the derivative of frequency-power functional relation. The chapter discusses the runtime estimation of the frequency-power derivative, and analyzes its accuracy using simulation. Moreover, it analyzes the convergence and stability properties of the control law, and formally shows that rapid settling time can be attained under wide range of derivative-estimation errors. Using a cycle-level multicore simulator and industry-standard benchmarks, the proposed control law is tested under various workloads and core microarchitectures, and was found to yield faster settling time than offline-designed fixed-gain controllers. Chapter 5 discusses the use of the core-power control law in conjunction with an iterative optimization algorithm to solve the problem of maximizing chip performance subject to chip-level power budget constraints.

3.2 Control Law

Consider the discrete-time scalar system shown in Figure 7, where the plant is modeled as a memoryless, time-varying nonlinear system of the form $P = g_n(\phi)$; n denotes

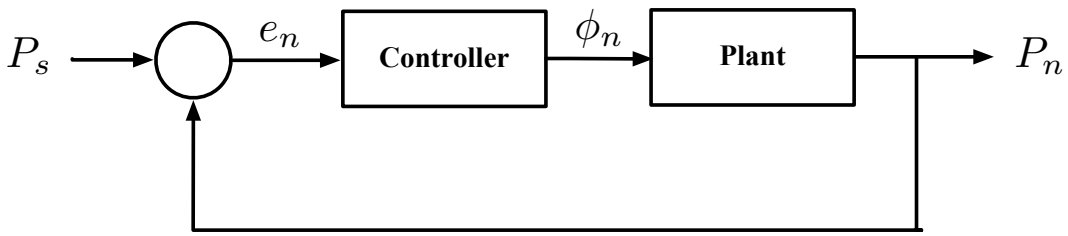


Figure 7: Power control system

(discrete) time and $g_n : R \to R$ is the function defining the system at time n. Let P_s be a given reference input, and suppose that the purpose of the controller is to regulate the output in the sense that $\lim_{n\to\infty} P_n = P_s$. To this end we use an integral controller of the form

$$\phi_n \;=\; \phi_{n-1} + K_n e_{n-1} \tag{1}$$

for a suitable gain $K_n > 0$, where the plant is defined as

$$P_n \;=\; g_n(\phi_n), \tag{2}$$

and it is evident from Figure 7 that

$$e_n \;=\; P_s - P_n, \tag{3}$$

for all $n = 1, \ldots$. Thus, once the gains K_n, $n = 1, 2, \ldots$ are specified, Equations (1)-(3) define the closed-loop system in a recursive manner. Suppose that at time n the function $g_n(\cdot)$ is known, we have a measurement of the control signal ϕ_{n-1}, and are able to compute the derivative term $g_n'(\phi_{n-1})$. We set the gain K_n to the following value,

$$K_n \;=\; \frac{1}{g_n'(\phi_{n-1})}. \tag{4}$$

We point out that if the plant is time invariant, namely $g_n(\cdot) = g(\cdot)$, then the recursive computation of e_n, defined by Equations (1) - (4), effectively is Newton's

method for finding a zero of the equation $e(\phi) = P_s - g(\phi) = 0$. In this case, we have the following well-known result: there exists a positive constant $0 < \beta < 1$ such that for every $n = 1, 2, \ldots$,

1. If $e_{n-1} \geq 0$ then

$$e_n \leq 0. \tag{5}$$

2. If $e_{n-1} \leq 0$ then

$$\beta e_{n-1} \ \leq \ e_n \ \leq \ 0. \tag{6}$$

This implies that $|e_n| \leq A\beta^n$ for some $A > 0$ for $n = 3, 4, \ldots$, and hence the error term e_n converges exponentially-fast to zero. Consider now the time-varying case, where the closed-loop system is defined via Equations (1) - (4). It can be shown that a positive constant $0 < \beta < 1$ exists such that for every $n = 1, 2, \ldots$,

1. If $e_{n-1} \geq 0$, then

$$e_n \ \leq \ g_{n-1}(\phi_{n-1}) - g_n(\phi_{n-1}). \tag{7}$$

2. If $e_{n-1} \leq 0$, then

$$\begin{aligned} &\beta e_{n-1} + \big(g_{n-1}(\phi_{n-1}) - g_n(\phi_{n-1})\big) \\ &\leq \ e_n \ \leq \ g_{n-1}(\phi_{n-1}) - g_n(\phi_{n-1}). \end{aligned} \tag{8}$$

Proofs of the above inequalities are included in Appendix A. Equations (7)-(8) imply that under a time-varying plant, P_n converges exponentially-fast toward a band (tolerance) around the target level P_s, and the width of the band depends on how fast the plant varies. The next section discusses the estimation of the frequency-power derivative $g'_n(\phi_n)$.

3.3 Frequency-Power Derivative Estimation

Consider a core with a frequency ϕ and a supply voltage V. The power dissipation at the core is a function of both the voltage and the frequency, as well as the workload. Denoted by $P(\phi, V, t)$, it has the following form,

$$P(\phi, V, t) = \alpha(t)CV^2\phi + P_L(V, t). \tag{9}$$

This equation, derived from basic physical principles, has been established in the literature; see, *e.g.*, [47, 4]. The first term in its right-hand side (RHS), $\alpha(t)CV^2\phi$, is the dynamic-power component resulting from the switching activity, and the second term, P_L, is the leakage power. The term $\alpha(t)$ is a time-varying workload parameter representing the switching activity of the processor's logic gates, and C is the total processor capacitive load. The leakage power P_L depends on the voltage setting and the processor temperature. Equation (9) presents an incentive for selecting low supply voltages, since P depends on V in a quadratic fashion. However, there exists a frequency-dependent bound on how low V can be set. Reducing the supply voltage of CMOS circuits generally increases their propagation delay [48], and this may violate timing constraints requiring all propagation delays to be less than the clock period $\theta = \frac{1}{\phi}$. Therefore, manufacturers specify a mapping $V(\phi)$, determined at design time, to guide the selection of voltage levels as a function of frequency. In light of the voltage-frequency dependence, Equation (9) can be re-expressed as

$$P(\phi, t) = \alpha(t)CV(\phi)^2\phi + P_L(V(\phi), t), \tag{10}$$

and its derivative is given by

$$\frac{dP(\phi, t)}{d\phi} = \alpha(t)C\Big(V(\phi)^2 + 2V(\phi)\ \phi\ \frac{dV}{d\phi}\Big) + \frac{dP_L}{dV}\frac{dV}{d\phi}. \tag{11}$$

The utility of power-derivative estimation is especially attained at run-time, where it can be implemented as a component of power-control schemes. Typically, these

schemes operate at low levels that favor low-complexity implementations, and are invoked with periods in the range of $2 - 40$ milliseconds [24]. The power derivative is amenable to a real-time implementation upon approximations described in the sequel. First, the RHS of the gradient (the leakage-power component) is dropped given the dominance of the dynamic power both in terms of the magnitude and rate of variations. Innovations in VLSI manufacturing, most notably high-κ dielectrics [49], have reduced leakage power to 18% of the total power consumed by the Intel Dunnington processor [50]. The contribution is reported to have been reduced further in recent processors such as Intel Westmere [51]). Moreover, temperature-induced variations in leakage power occur at a much slower rate compared to those induced by the workload. Therefore, relative to the dynamic-power term, the leakage power can be treated as a constant. Second, the mapping between voltage and frequency is assumed to be well approximated by an affine function of the form

$$V(\phi) \;=\; m\phi + V_0, \tag{12}$$

which can be numerically determined offline using the voltage-frequency values acquired from the manufacturer's data sheet. The voltage setting in general increases with frequency (please see [20, 37] for empirical evidence), yielding a strictly positive power gradient. Direct estimation of the activity factor $\alpha(t)$ is nontrivial. However, assuming $P(\phi, t)$ and P_L can be measured at runtime, the power derivative can be re-expressed as

$$\frac{dP(\phi,t)}{d\phi} = \big(P(\phi,t) - P_L\big)\left(\frac{1}{\phi} + \frac{2}{V(\phi)}\frac{dV(\phi)}{d\phi}\right). \tag{13}$$

Substituting Equation (12) into the above yields

$$\frac{dP(\phi,t)}{d\phi} = \big(P(\phi,t) - P_L\big)\left(\frac{1}{\phi} + \frac{2m}{m\phi + V_0}\right). \tag{14}$$

Implementing the above estimator at runtime requires measuring the total power and the leakage power of the core; a capability that is readily available in commercial processors such as Intel Sandy Bridge [52]. To assess the accuracy of the power derivative estimator, we use the relative error $\epsilon_P(\phi)$ which is defined as:

$$\epsilon_P(\phi) = \frac{\Big|\Delta P(\phi) - \Delta\phi P'(\phi)\Big|}{\Big|\Delta P(\phi)\Big|}, \tag{15}$$

where $\Delta P(\phi) = P(\phi + \Delta\phi) - P(\phi)$ is the change in power upon perturbing the frequency ϕ by a small amount $\Delta\phi$, which is acquired via simulation, and $\Delta\phi P'(\phi)$ is the *predicted* change in power acquired via linear interpolation using the power derivative $P'(\phi)$. The relative error was assessed via simulation using a setup described in the next section. Figure 8 plots the relative error of the power-derivative estimator for the simulated programs, which are selected from the industry-standard SPEC2006 benchmark suit. The plotted relative error is the average of three experiments performed with different frequency values $\phi \in [2.1, 2.4, 2.7]$ GHz, with a frequency perturbation of $\Delta\phi = 50$ MHz. The observed errors range from 1.8% to 7.8%, with an average of 4.11%.

3.4 Tracking Results

The proposed power control algorithm is evaluated using *Zesto* [53]; a detailed X86 microprocessor simulator, which is integrated with the McPAT tool [54] for microarchitectural power modeling. The simulated configuration, which is shown in Fig. 9, is an asymmetric multicore processor consisting of two in-order cores and two out-of-order cores, where the core are equipped with private L1 and L2 cache memories, communicate via a 2×2 mesh network. The configuration of the cores and cache memories are listed in Table 1. The simulated programs are drawn from the industry-standard SPEC2006 benchmark suit. Detailed simulation is performed for

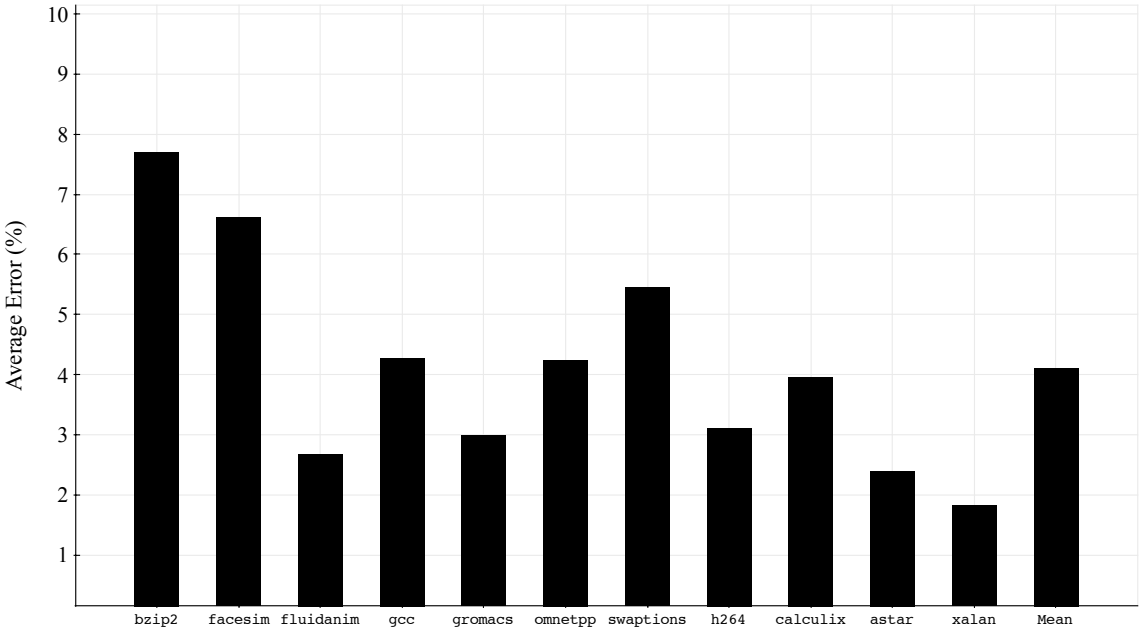


Figure 8: Error analysis of the power-gradient estimator

a period of 100×10^6 instructions, and is preceded by fastforwarding 1×10^9 instruction to warm up the processor and memory state. The control period of the regulation algorithm is set to 5×10^{-3} seconds. The proposed adaptive-gain controller is compared to fixed-gain integral controllers whose gain is drawn from the set $K = [25, 50, 75, 100, 150, 270, 385, 500]$. Controllers are evaluated based on their settling time, which is the time taken to reach within $\pm 5\%$ of the power set-point. The initial frequency and supply voltage for each tracking experiment is set to 3GHz and 0.9V, respectively.

First, we present tracking results using an out-of-order core driven by programs of varying behavior. In this setting, the power profile is strongly dependent on the activity factor $\alpha(t)$ intrinsic to the workload. High switching activity is generally associated with compute-intensive programs that have high instruction-level parallelism (ILP) and limited memory accesses, resulting in high core utilization. On the other hand, low-ILP and memory-bound programs experience lower core utilizations that yield lower switching activity. Controller-gain choice is therefore strongly dictated by

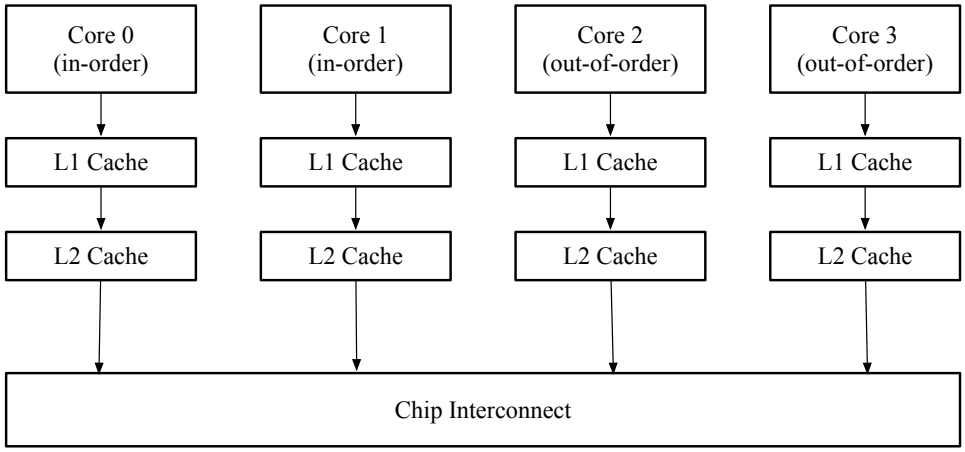


Figure 9: Simulated Asymmetric Multicore Processor Configuration

Table 1: Machine configuration for power tracking

Parameters	Out-of-order Core	In-order Core
Architectural Configuration		
ISA	x86 IA32	
Pipeline Depth	20 stages	16 stages
Fetch/Decode	4 instructions	2 instructions
Execution	6 ports	3 ports
L1 Cache	4-way 32KB	4-way 32KB
L2 Cache	8-way 512KB	8-way 512KB
Physical Configuration		
Clock Frequency	1.85-3.75GHz	
Supply Voltage	0.6-1.0V	
Feature Size	45nm	

the workload as will be shown in the sequel. Consider the tracking results plotted in Figure 10(a) and 10(b) for the programs *deal* and *omnetpp* respectively. The initial frequency ϕ_0 is set to $\phi_{max} = 3 \times 10^9$ Hz, and the set-point is chosen as $P_s = P(\phi_{max})$. Under both programs, the proposed controller adapted the gain to achieve a rapid settling time of 3 control periods (15 milliseconds). Comparable settling times were observed when the gain was fixed at 75 and 150 for *deal* and *omnetpp* respectively. The workload-induced variation in the static gains underscores the disadvantage of offline-designed controllers, since selecting a gain of 75 would severely degrade the settling time in *omnetpp*, whereas selecting 150 would cause oscillations for *deal*.

The advantages of the adaptive-gain controller are observed over a wider benchmark set as summarized in Figure 11. The average settling time of the adaptive algorithm is 0.0108 seconds, which is approximately 50.7% of the fastest static settling time (0.0217 seconds at $K = 150$).

Next, the strong relationship between the controller gain and the the underlying core microarchitecture is illustrated by executing the same program on the asymmetric multicore configuration shown in Fig. 9. The out-of-order core employs aggressive pipelining and speculation to exploit instruction-level parallelism, yielding a higher number of instructions executed per cycle (IPC) and power compared to the less-sophisticated in-order core. Figures (12-14) show the power-tracking results of the asymmetric multicore chip, using benchmark *milc* for adaptive gain, high fixed gain ($K = 500$), and (c) low fixed gain ($K = 25$) controllers. Each core executed the same benchmark and the execution was partitioned into three phases. For each phase the power set-point was changed for each core as shown in Table 2. The power budgets are shown as dotted lines in the figure. We can observe how well the adaptive gain and static gain controllers track and maintain new power budgets.

Table 2: Set-points for multicore power tracking

Core	Phase 1	Phase 2	Phase 3
Core_0 (in-order)	6.5 W	5.5W	7.5W
Core_1 (in-order)	6.5 W	7.5W	5.5W
Core_2 (out-of-order)	12 W	10W	12W
Core_3 (out-of-order)	12 W	14W	12W

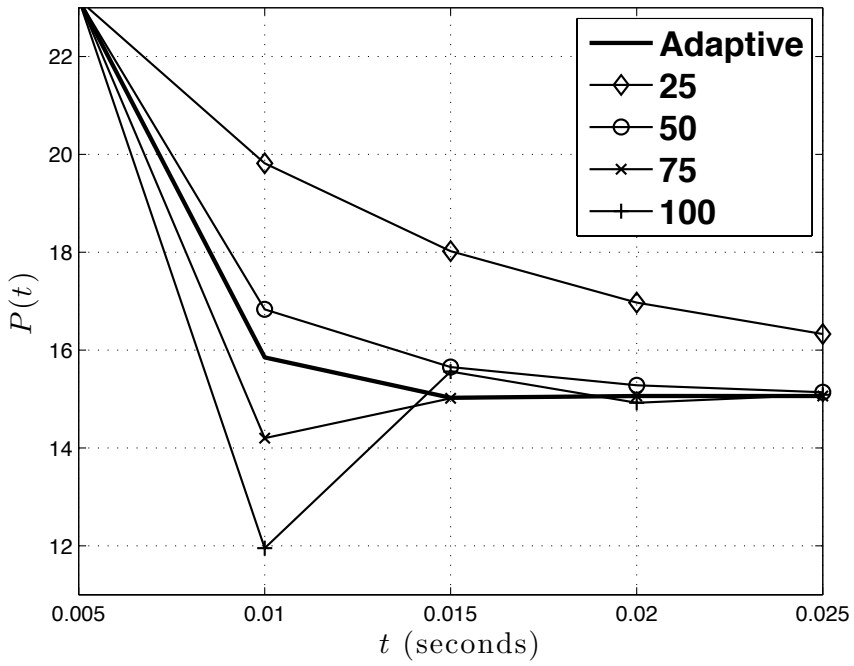
The adaptive gain controller tracked the varying reference signals within 3 control periods (15 ms) for both in-order and out-of-order cores. The high fixed gain controller is as effective as the adaptive gain controller for the in-order cores but inefficient for the out-of-order cores. The performance difference is caused by the microarchitectural difference between the two cores. Under the same workload, $\frac{dP}{d\phi}$ is greater in the out-of-order case, since it has a higher capacitance C, and can execute more instructions per unit time (larger $\alpha(t)$) compared to the in-order processor. When the power budget

is increased, the out-of-order processor requires a smaller frequency correction Ke_n compared to the in-order case by virtue of its steeper power vs. frequency relationship. Thus, it is expected that a high-enough gains may cause significant overshoot and even oscillations in the out-of-order case while being beneficial in the in-order case as shown in Figures (12 - 14).

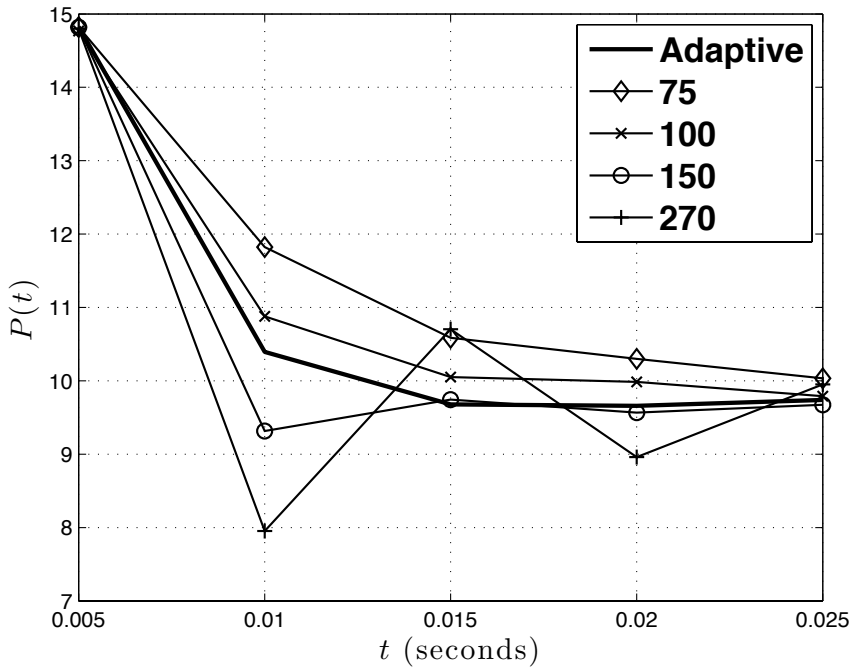
3.5 Related Work

[46] proposed a proportional controller to regulate the power of blade servers. The plant was modeled as a linear, memoryless system whose gain was found offline via offline profiling. The control algorithm adjusted the processor DVFS setting using the error (difference between the power-reference value and the actual power) scaled by the reciprocal of the plant gain. Error integration is carried out as part of the actuator; a first-order sigma-delta modulator, to achieve zero steady-state error. The paper analyzed the convergence and robustness of the algorithm under errors in plant modeling. Compared to [46], the control law proposed in this chapter regulates the power of cores as opposed to blade servers. Moreover, it assumes a more general (convex) plant and is formally shown to achieve achieve rapid settling time.

[22, 23] proposed a scheme where the power budget of a chip multiprocessor is partitioned between voltage islands to maximize performance. The power fractions assigned to each voltage island are tracked using a proportional-integral-derivative controller (PID), which is designed using an offline-generated *average* linear system model. The reliance on average offline profiling renders the approach prone to instability if the runtime plant deviates from the offline plant. In contrast, the stability and rapid tracking of the proposed control law is formally shown, as well as its robustness to derivative estimation errors.



(a) *dealII*



(b) *omnetpp*

Figure 10: Power tracking for *dealII* and *omnetpp*

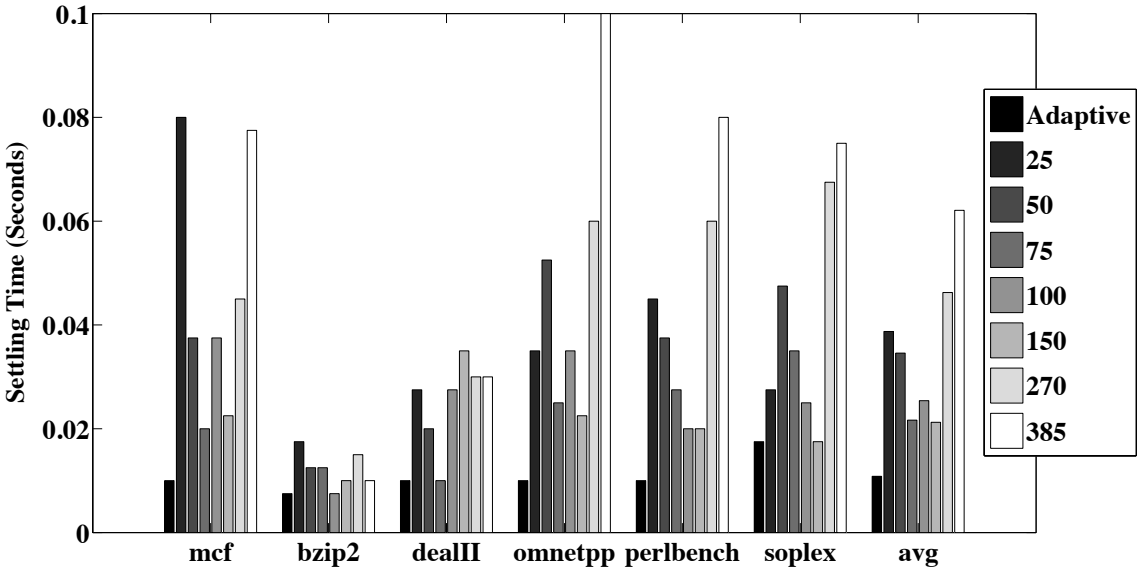


Figure 11: Settling-time comparison

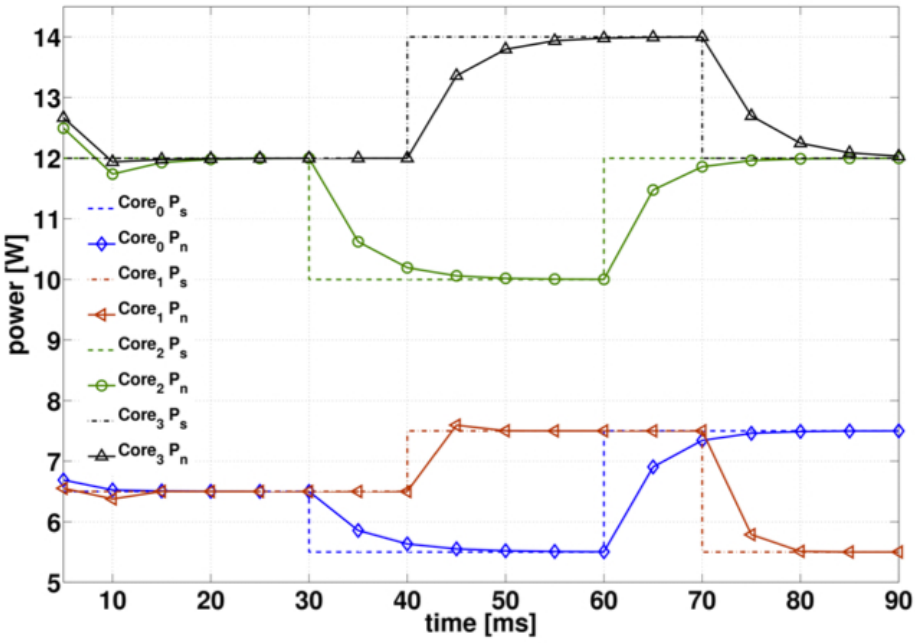


Figure 12: Adaptive gain controller

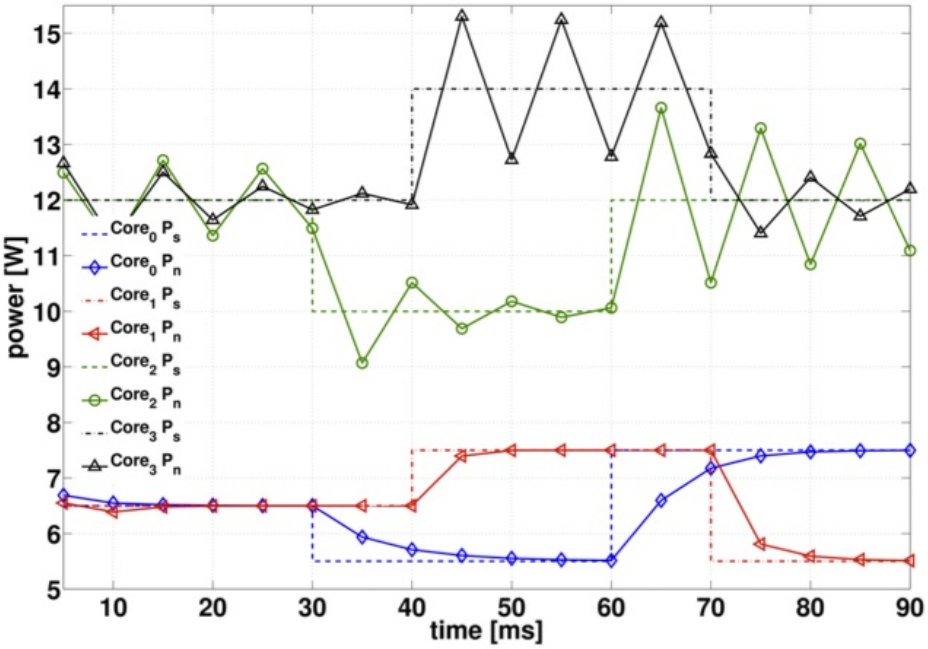


Figure 13: High fixed gain controller ($K = 500$)

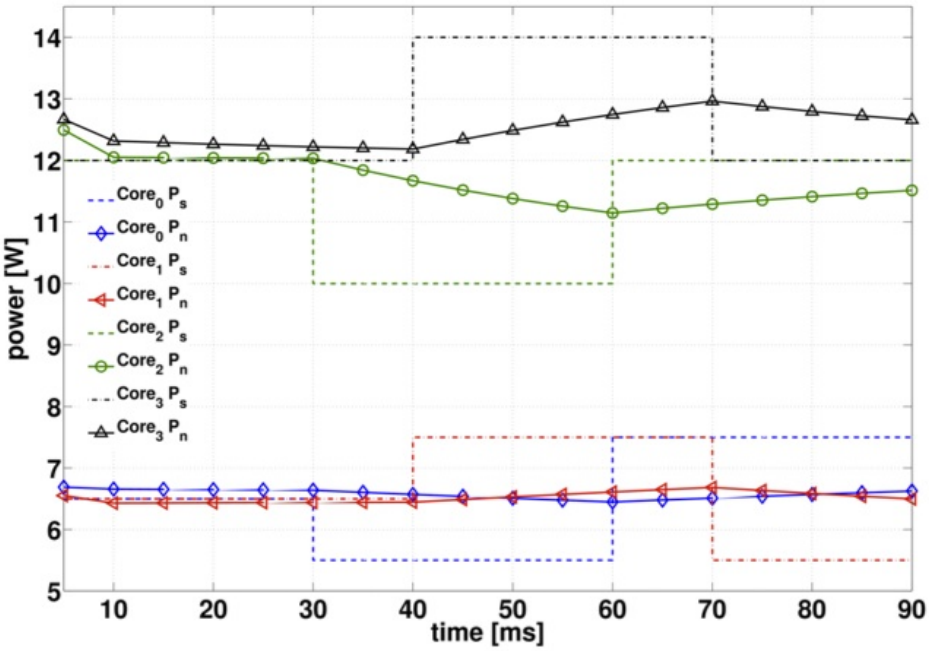


Figure 14: Low fixed gain controller ($K = 25$)

CHAPTER IV

THROUGHPUT REGULATION

4.1 Overview

Throughput regulation has been proposed to improve the predictability in real-time embedded systems. In this setting, *a-priori* guarantees on task completion times are required prior to implementation, which were traditionally acquired using worst case execution time (WCET) analysis [55]. The drawback of this approach is that the WCET bounds are conservative; peak performance is significantly reduced and the bounds are rarely approached. Consequently, the use of WCET analysis has generally been limited to in-order cores without caches. The successful application to high performance out-of-order cores is more challenging. For example, as shown in [31], execution-time uncertainty increases significantly in out-of-order processors attributed to the use of speculation in the control path compounded by variability in the memory hierarchy.

An early example of throughput regulation using DVFS is [30], which was geared to improve the energy-efficiency of hard real-time embedded systems. Recently, Suh *et. al.* [31] proposed regulating the throughput of embedded out-of-order processors using feedback control. The proposed algorithm is a proportional-integral-derivative (PID) controller that adjusts the DVFS setting of the processor to track a desired throughput level. The parameters of the controller (values of the gain of the proportional, integral, and derivative components) are calculated offline. The authors report reasonable tracking under minimal deviations between the offline and runtime plant.

The approach proposed in the next section is based on an integral controller whose gain is adjusted online to achieve rapid throughput regulation under a variable

workload. The gain computation is based on the sample derivative of the frequency-throughput functional relation, estimated using infinitesimal perturbation analysis (IPA).

4.2 Control Law

Consider the control system shown in Figure 15, where the output signal ζ_n denotes the instruction throughput, measured as $\zeta_n = \frac{M_n}{T_c}$, where M_n is the number of instructions committed in the n-th control period, and T_c is the length of the control period.

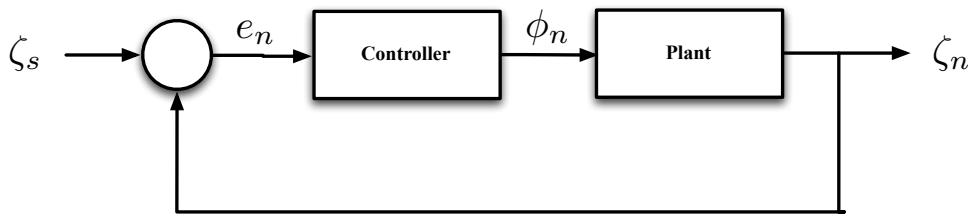


Figure 15: Throughput control system

The reference input ζ_s is the target throughput, and the control variable ϕ_n is the clock frequency. The system shown in Figure 15 pertains to an instruction throughput regulator that can be implemented at each individual core in a many-core processor. Similar to the power control law proposed in chapter 3, we use an integral controller defined as

$$\phi_n = \phi_{n-1} + K_n e_{n-1}, \tag{16}$$

where $K_n = \frac{1}{\zeta_n'(\phi)}$ is a time-dependent gain calculated using the sample derivative of the frequency-throughput functional relation. The next section presents a brief overview of out-of-order execution followed by a queuing model that captures its semantics. The sample derivatives of the execution time and throughput of instructions are then derived using IPA.

4.3 Performance Sensitivity Analysis

Consider the processor model shown in Figure 16, which represents a generic design resembling most commercial processors [56]. The instruction-delivery subsystem performs the tasks of instruction fetch and buffering, as well as branch prediction. Fetched instructions enter a decode pipeline, after which are *dispatched*. During the dispatch stage, instruction registers are renamed, and they are allocated an entry in (i) the issue buffer (whose entries are called reservation stations), and (ii) the reorder buffer (ROB). Issue-buffer entries hold the operation and operands necessary to compute it. Upon the availability of all operands and an appropriate execution unit, instructions are *issued*, or sent to their respective functional unit for execution. Results are sent back from the functional unit to the reservation stations and to the ROB entry until it is hazard-free for the results to be written into the register file (RF).

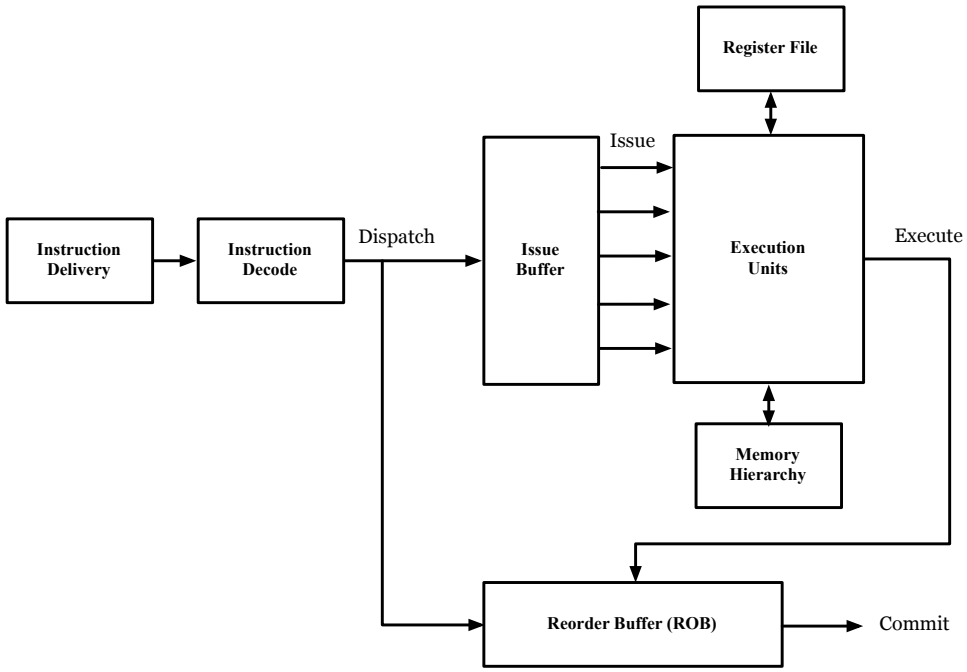


Figure 16: A generic out-of-order processor

Following execution, instructions can commit (leave the ROB, also called retire or graduate) in the same order they were dispatched. This departure-order constraint,

denoted as *in-order commit*, effectively makes the program behave as if executing on a simple in-order pipeline composed of four primary units [6] as shown in Figure 17. This allows dependences to be tracked and correct state to be recovered in the case of trap or a branch misprediction, and confers instruction-level parallelism (ILP) gains by allowing instructions to execute out-of-order.

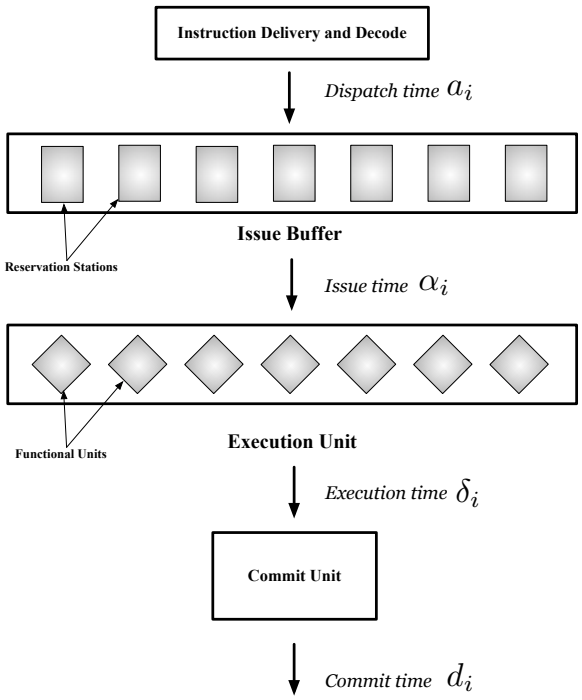


Figure 17: Primary units of out-of-order execution

We rely on *interval analysis* proposed by Eyerman *et. al.* [56] to intuitively explain the relationship between the clock frequency and the execution time . Interval analysis focuses on the flow of instructions through the pipeline and the events that disrupts it, which are called *miss events*. Figure 18 illustrates the concept, where the x-axis represents Time, and the y-axis represents the instructions committed per cycle (IPC), which alternate between zero and a maximum value representing the commit width. An interval starts with a period of smooth flow of instructions followed by a miss event. The beginning of the next interval is marked by the instant when instructions start flowing smoothly again. The periods of smooth-instruction flow encapsulate execution

in the core's functional units and transitions between its pipeline stages, and their durations are certainly a function of the core's clock frequency. The dependence on the clock frequency also applies to miss events serviced inside the core; *e.g.* cache and table-lookaside buffer (TLB) misses and branch mis-predictions. Conversely, the duration of miss events processed in another clock domain; mainly load requests processed in dynamic random-access memory (DRAM), is independent of the clock frequency.

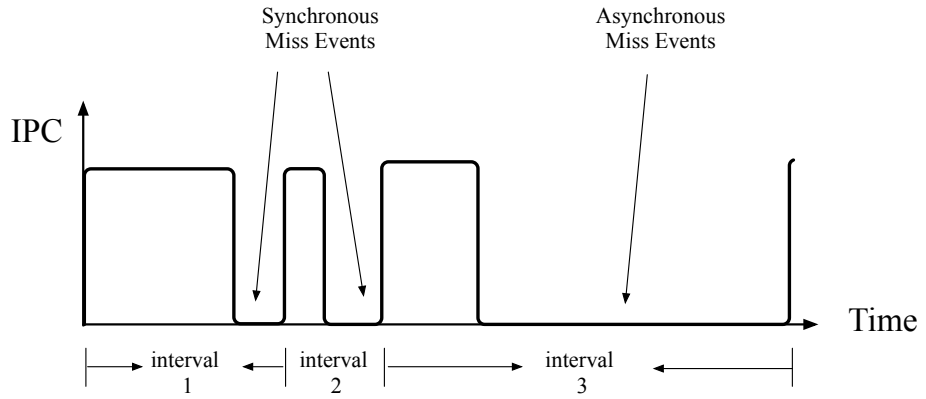


Figure 18: Interval analysis of processor execution

We call any event processed outside the core *asynchronous*. Otherwise, the event is called *synchronous*. Referring, to Figure 18, changing the clock frequency would alter (stretch or compress) the whole region except for the asynchronous miss event of interval 3. Suppose that the cycle count at the onset of asynchronous miss-event is C. It can be seen that the slope of the execution time is equal to C, since increasing the clock period $\theta \rightarrow \theta + \Delta\theta$ affects only the synchronous region, whose execution time grows by $C\Delta\theta$. Next, we present a queuing model of an OOO model that formally captures these intuitive facts. Consider an instruction sequence $I_i, i = 1, 2, ...$ to be executed by a core operating with a frequency $\phi = \frac{1}{\theta}$. The following model is constructed with reference to simplified processor model shown in Figure 17. Let us denote the arrival (dispatch) time of instruction i as a_i, and by α_i the time i is issued and starts executing at its designated functional unit. Furthermore, denote by δ_i the

time i completes execution, and by d_i the commit (departure) time. The dispatch time can be expressed as:

$$a_i = \ell(i)\theta, \tag{17}$$

where $\ell(i)$ is an the cycle count at the dispatch instant. In practice, a_i may contain asynchronous events caused by instruction-cache misses, but are dropped given their negligible contribution to execution time. Moreover, we make the approximation that finite issue buffer/ROB capacities do not cause dispatch stalls. Next, consider the issue time of instruction i, denoted as $\alpha(i)$. Assuming unlimited functional units, if all of instruction i's operands are ready upon dispatch, it can be issued in the next clock cycle, in other words $\alpha_i = a_i + \theta$. If on the other hand an operand is not ready, i can be dispatched in the cycle following operand readiness. Denote by $k(i)$ the index of the instruction producing the operand. Accordingly, $\alpha_i = \delta_{k(i)} + \theta$, and the issue time is given by:

$$\alpha_i = \max\{a_i, \delta_{k(i)}\} + \theta. \tag{18}$$

Next, consider the execution time of instruction i. Instructions whose execution is comprised only of synchronous events are called synchronous; these include arithmetic operations and memory instructions that hit in the core cache. Otherwise, instructions are denoted as asynchronous. Observe that asynchronous instructions in practice may contain a synchronous execution component attributed to address generation and core-cache access latency. This can be dealt with by treating the synchronous component as a separate instruction whose output will be an operand to the asynchronous component. The execution latency of synchronous instructions is generally of the form $n(i)\theta$ where $n(i)$ is an integer dependent on the type of instruction. On the other hand, since the execution of asynchronous instruction is carried out in a different clock domain, its execution latency can be represented as a constant T_{mem} independent of θ. Hence, an approximate equation for the execution time δ_i is

given by:

$$\delta_i = \begin{cases} \alpha_i + n(i)\theta : & i \text{ is synchronous} \\ \alpha_i + T_{mem} : & i \text{ is asynchronous} \end{cases} \tag{19}$$

Finally, the departure (commit) time can be expressed as:

$$d_i = \max\{\delta_i, d_{i-1}\} + \theta \tag{20}$$

To define the IPA derivative , first define $v(i)$ as

$$v(i) = \begin{cases} 0 & : i \text{ is synchronous} \\ n(i) & : i \text{ is asynchronous,} \end{cases} \tag{21}$$

and

$$m(i) := \max\{m \leq i : I_m \text{ arrived upon completion}\}. \tag{22}$$

Proposition 1. *The following equations (23) and (24) are in force for all* $i = 1 \ldots M$.

$$\alpha_i'(\theta) = \begin{cases} \alpha_{k(i)}'(\theta) + v(k(i)) + 1 & : I_i \textit{ stalled by a data dependency} \\ \ell(i) + 1 & : \textit{otherwise.} \end{cases} \tag{23}$$

and

$$d_i'(\theta) = \alpha_{m(i)}'(\theta) + v(m(i)) + i - m(i) + 1, \tag{24}$$

Proof. The above follows by direct application of equations (17) - (20). First, consider equation (23). If I_i issues upon arrival, then by (18) $\alpha_i = a_i + \theta$. By (17), and taking the derivative, $\alpha_i' = \ell(i) + 1$. This is the second case of (23).

On the other hand, suppose I_i stalls upon arrival due to a data dependency. By (18), $\alpha_i = \delta_{k(i)} + \theta$. Now there are two sub-cases: (i) $I_{k(i)}$ is a synchronous instruction, and (ii) $I_{k(i)}$ is an asynchronous memory instruction. In sub-case (i), by (19) we have that $\delta_{k(i)} = \alpha_{k(i)} + n(k(i))\theta$, hence $\alpha_i = \alpha_{k(i)} + n((k(i)+1)\theta$; consequently $\alpha'_i = \alpha'_{k(i)} + n(k(i) + 1$, which is the first case of (23).

In sub-case (ii), (19) implies that $\delta_{k(i)} = \alpha_{k(i)} + T_{mem}$, hence $\alpha_i = \alpha_{k(i)} + T_{mem} + \theta$. Consequently, $\alpha'_i(\theta) = \alpha'_{k(\theta)} + 1$, which is the first case of (23). This establishes equation (23) under all situations.

Next, consider Equation (24). By (20), if I_i is stalled after it's execution due to in-order commit enforcement, then $d_i = d_{i-1} + \theta$, otherwise $d_i = \delta_i + \theta$. Therefore, we have that $d_i = d_{m(i)} + (i - m(i))\theta = \delta_{m(i)} + (i - m(i) + 1)\theta$. Hence $d'_i(\theta) = \delta'_i(\theta) + (i - m(i) + 1)$. By (19), $\delta'_{m(i)}(\theta) = \alpha'_{m(i)}(\theta) + v(m(i))$, from which (24) follows.

□

Direct implementation of the recursive IPA derivative in (24) requires extensive tracking of instruction timing and dependencies and is likely to yield a complex implementation. Instead, we opt for an equivalent implementation where the IPA estimator is calculated by analyzing the inter-departure time of instructions. Define $\xi(.)$ as

$$\forall x = A\theta + BT_{mem} \quad \rightarrow \quad \xi(x) = A. \tag{25}$$

This function serves as a synchronous-cycles extractor. In other words, if an observed duration contains an additive mix of synchronous and asynchronous events, $\xi(.)$ returns the number of synchronous cycles. We envision calculating the IPA derivative using a hardware implementation of the function $\xi(.)$. The circuit observes the inter-departure time of instructions $\Delta d_i = d_i - d_{i-1}$, and calculates the IPA derivative as

$$d'_i(\theta) = d'_{i-1}(\theta) + \xi\Big(\Delta d_i\Big). \tag{26}$$

Using $\theta = \frac{1}{\phi}$, we can express the departure time derivative with respect to the frequency as

$$d'_i(\phi) = \frac{-d'_i(\theta)}{\phi^2}. \tag{27}$$

Another performance measure of interest is the throughput, measuring the number of instructions committed per unit time. If we denote by M the instruction length of the program, the throughput is given by

$$\zeta(\phi) = \frac{M}{d_M(\phi)}, \tag{28}$$

and the sample throughput derivative is expressed as

$$\zeta'(\phi) = d'_M(\theta)\frac{M}{\phi^2 d_m(\phi)^2}\ . \tag{29}$$

To assess the accuracy of the sample throughput estimator , we use the relative error metric defined as:

$$\epsilon_\zeta(\phi) = \frac{\Big|\Delta\zeta(\phi) - \Delta\phi\zeta'(\phi)\Big|}{\Big|\Delta\zeta(\phi)\Big|}, \tag{30}$$

where $\Delta\zeta(\phi) = \zeta(\phi+\Delta\phi) - \zeta(\phi)$ is the change in throughput upon perturbing the frequency ϕ by a small amount $\Delta\phi$, which is acquired via simulation, and $\Delta\phi\zeta'(\phi)$ is the *predicted* change in throughput acquired via linear interpolation using the sample throughput derivative $\zeta'(\phi)$. The relative error was assessed using a simulation setup described in the next section and industry-standard SPEC2006 and PARSEC benchmarks. The relative error is summarized in Table 3. Across benchmarks, we

Table 3: Error analysis of the IPA estimator

Benchmark	**Departure time** (%)	**Throughput** (%)
gcc-scilab	4.4	5.72
gromacs	5.15	5.85
deal	0.40	9.52
omnetpp	4.21	5.24
calculix	1.33	1.84
bzip2-chicken	5.21	6.00
calculix	1.33	1.84
xalan	10.48	12.39
hmmer-retro	4.07	4.57
h264	0.44	0.95
perl-diffmail	7.46	8.9
lbm	8.6	11.44
fluidanimate	0.04	0.04
swaptions	1.72	2.79
facesim	15.81	17.86
sjeng	31.66	33.71

observe an average error of 7.76% for the departure-time sample derivative, and 6.69% for the sample throughput derivative.

4.4 Simulation Results

The proposed throughput tracking algorithm is simulated using *Zesto* [53]; a cycle-level microprocessor simulator. The simulated processor configuration is shown in Fig. 19, and it comprises four out-of-order cores connected via a ring network. The cores includes a private L1 cache, and are connected via the network to a shared L2 cache. The parameters of the core and cache configurations are listed in Table 4.

Table 4: Simulated processor configuration for throughput regulation

Parameter	Configuration Value
Instruction set Architecture	x86 IA32
Reorder Buffer Size	128 entries
Execution Width	6 ports
Core Clock	2.0 - 4.75 GHz
Network and shared L2 clock	2.0 GHz
L1 Cache	4-way 32B-line 16 KB
Shared L2 Cache	8-way 32B-line 256 KB

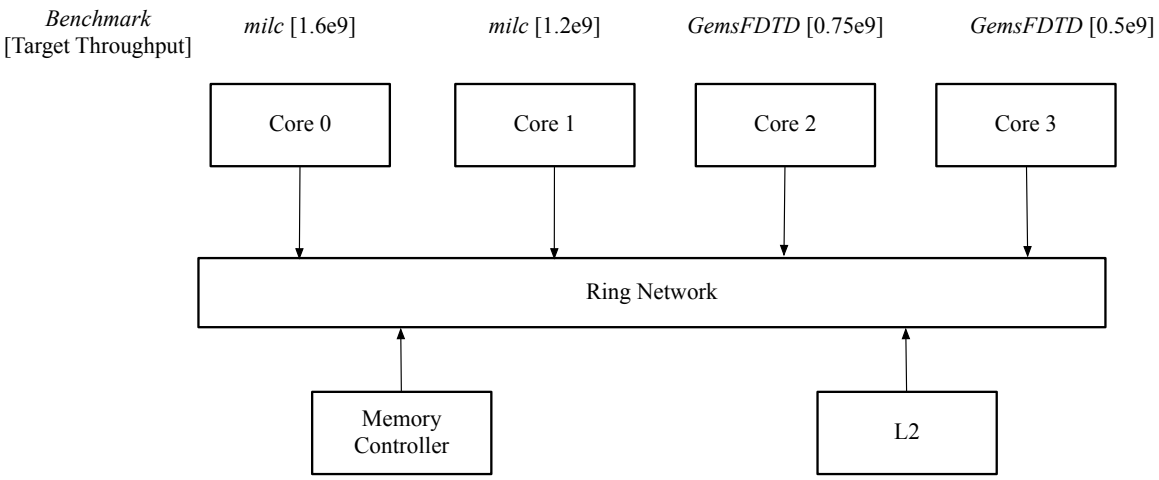


Figure 19: Simulated processor configuration for throughput tracking

The Control period is set to 100×10^3 instructions, and the core frequency range is between $2.0 - 4.75$ GHz. In the throughput tracking analysis, we chose two SPEC2006 benchmarks that can typically achieve two distinct levels of instruction throughput, milc and GemsFDTD. Two cores executed the milc benchmark with target throughput of 1.6×10^9 and 1.2×10^9 instructions per second, respectively, and the other cores executed the GemsFDTD benchmark with target throughput of 0.8×10^9 and 0.5×10^9 instructions per second, respectively. In all experiments, throughput remains unregulated for the first 2ms during which time each core executes at a constant frequency of 3.0GHz. At time $t = 2ms$ each core was assigned its target throughput and the throughput controllers where activated. The IPA controller is compared against controllers using a small fixed gain $Kn = 0.5$ and a large fixed gain $Kn = 5.0$. In Figure 20, we observe that the adaptive-gain IPA controller regulates the throughput and achieves tracking quite rapidly. In contrast, the small fixed-gain controller (Figure 21) is sluggish in regulating the throughput especially with the lower throughput benchmarks, while the large-gain controller (Figure 22) overshoots especially with the high throughput benchmark.

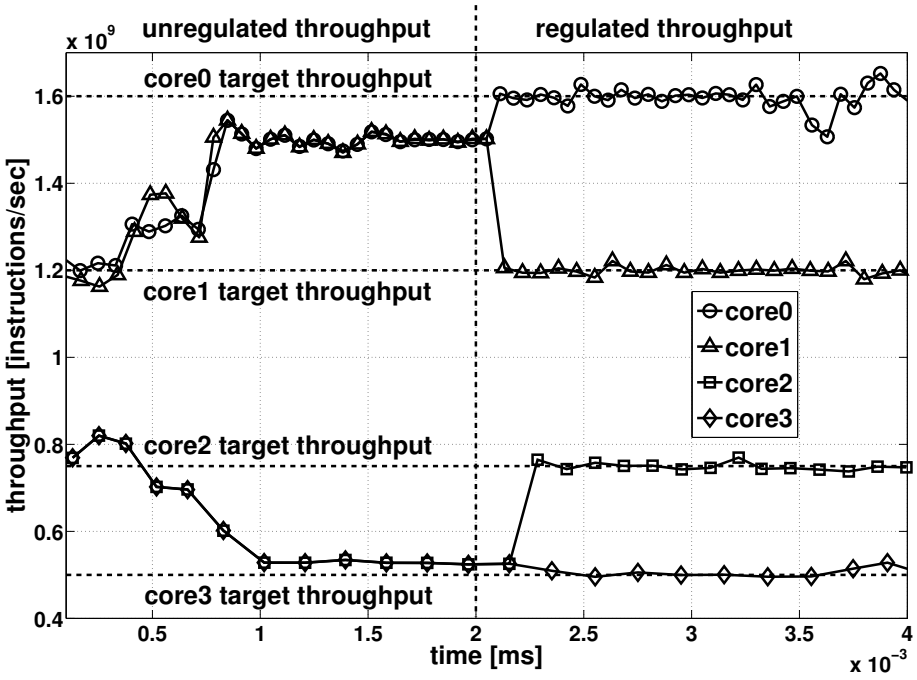


Figure 20: Throughput tracking using IPA

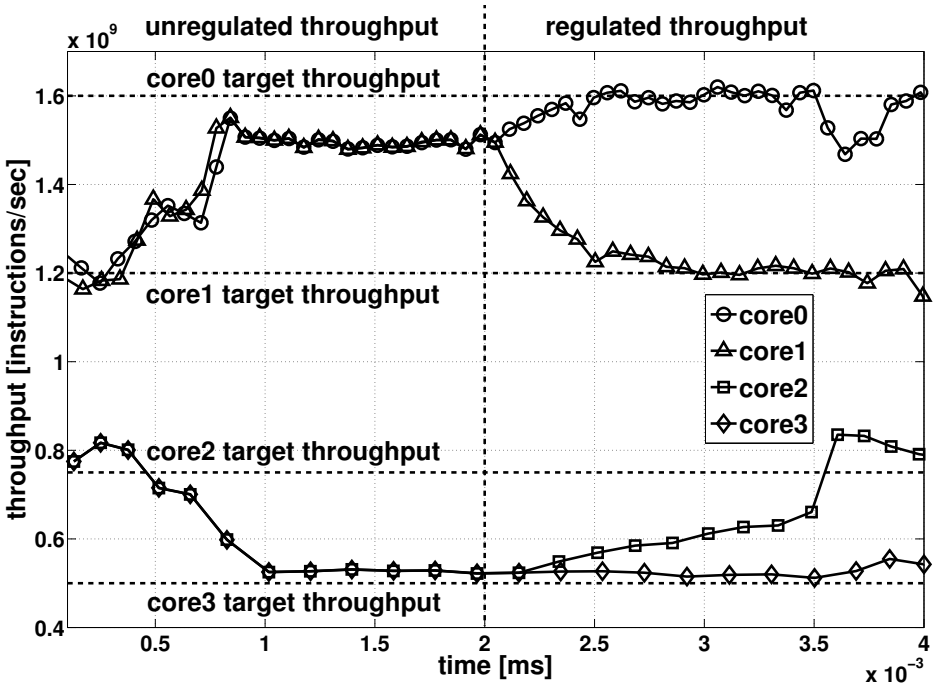


Figure 21: Fixed-gain throughput tracking ($Kn = 0.5$)

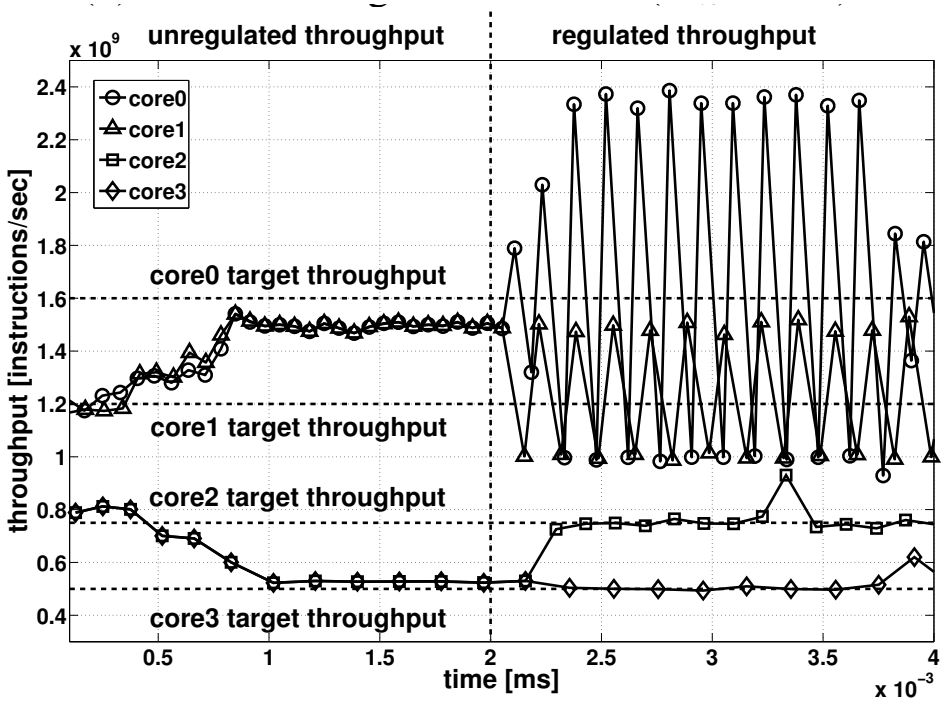


Figure 22: Fixed-gain throughput tracking ($Kn = 5.0$)

CHAPTER V

CHIP POWER CONTROL

5.1 Overview

The problem of chip power control is concerned with maximizing the performance of manycore processors while ensuring that the power envelope stays below the thermal design power (TDP), which reflects the capacity of the chip cooling system. Solving the chip power control problem has central implications on the design of the cooling and power supply systems of the processor [16, 4], as well as maximizing the performance returns of core scaling [57, 58].

The challenges to designing chip-power control schemes stem from 1) the limited computational resources available to implement the algorithms, and 2) the highly-varying relationships between the control variables, and the power and performance of the executing workloads. In this chapter, the chip power control problem is approached via decomposition to a *master* problem that is concerned with partitioning the TDP (chip-level power budget) between cores to maximize *fair speedup* [45], which is a performance metric that captures system-level performance as well fairness, and *regulation* subproblems that enforce the power portions (setpoints) assigned to each core. The regulation subproblems are solved using the core-power controllers proposed in Chapter 3. The master algorithm is a constrained-optimization algorithm based on the method of *gradient projection* [59], which updates the power setpoints iteratively using the sample power-throughput derivatives calculated at each core while adhering to power constraints. The master algorithm uses the derivative estimators described in Chapters 3 and 4 to estimate the feasible power setpoint range at each core, as well as estimating the peak application throughputs that are necessary

to maximize the fair speedup metric. The proposed chip power control algorithm is evaluated using a detailed multicore processor simulator and industry standard benchmarks from the SPEC2006 and PARSEC benchmark suites. Simulation results show that the proposed algorithm yields higher performance and power-accuracy than the state-of-the-art technique proposed in [14].

5.2 System Overview and Problem Definition

The setting of chip power control is illustrated in Fig. 23. The many-core processor is composed of N cores , each equipped with a frequency-voltage domain that can be independently configured from other cores. The frequency of the i-th core is denoted as ϕ_i, and $\Phi \in R^N$ denotes the vector of core frequencies $\Phi = [\phi_1\ \phi_2 \ldots\ \phi_N]^T$. Each core executes a single-threaded program; i.e. a multiprogrammed workload , that is independent from other cores. The power and throughput at the i-th core are functions of the workload and the core frequency setting, and are denoted as $P_i(\phi_i)$ and $\zeta_i(\phi_i)$, respectively.

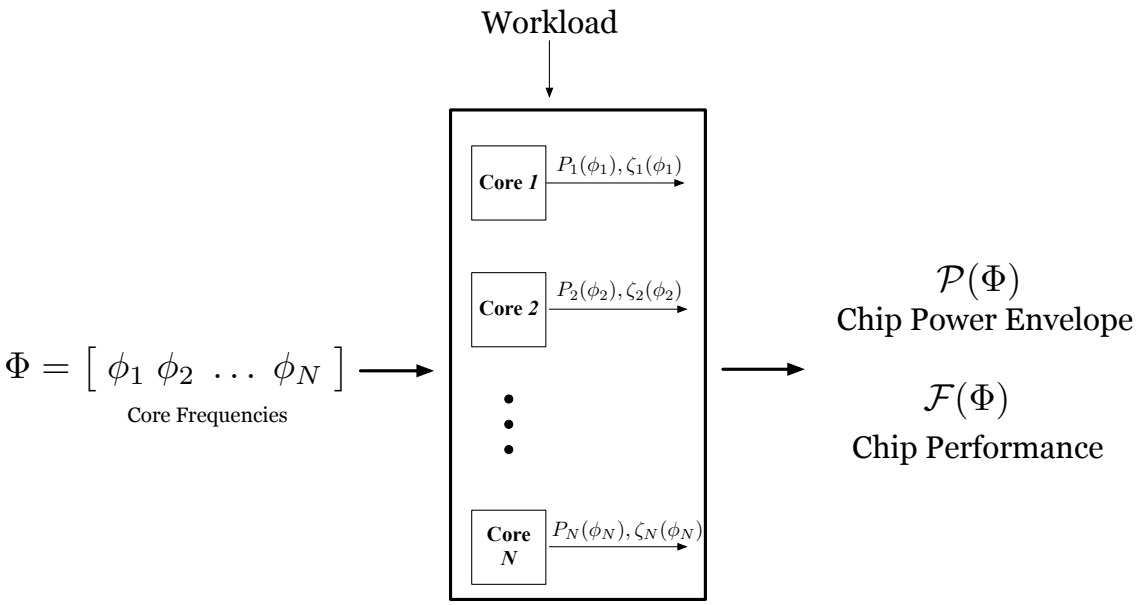


Figure 23: Chip Power Control Setting

The power envelope of the chip is modeled as $E(\Phi) = \mathcal{P}(\Phi) + P_c$, where $\mathcal{P}(\Phi)$

represents the power of the cores, given as:

$$\mathcal{P}(\Phi) = \sum_{i=1}^{N} P_i(\phi_i), \tag{31}$$

and P_c is the power of the remaining chip components; e.g. the interconnect and shared caches (if any). The frequency vector Φ influences how cores utilize other chip resources, and will have an indirect effect on P_c. The effect of this coupling is minimal in comparison to the core-power term $\mathcal{P}(\Phi)$, therefore, P_c is modeled as a constant, which can be measured at runtime since the power of the chip and the cores can be measured.

The proposed power control algorithm is concerned with maximizing the fair speedup metric, which was selected as chip-level performance measure by several proposals [44, 45, 14] since it provides a balance between system and application-level throughput. The fair speedup metric is the harmonic mean of *normalized* application throughputs, where the normalization reflects the impact of the power control algorithm on the application throughput regardless of their inherent throughput characteristics. The normalized throughput at the i-the core is defined as:

$$\rho_i(\phi_i) = \frac{\zeta_i(\phi_i)}{\zeta_i(\phi_{max})}, \tag{32}$$

where $\zeta_i(\phi_{max})$ is the normalization factor, representing the throughput when the core frequency is set to ϕ_{max} throughout execution. The fair speed up metric is defined as

$$\mathcal{F}(\Phi) = \frac{N}{\mathcal{G}(\Phi)}, \tag{33}$$

where the denominator $\mathcal{G}(\Phi)$ lis defined as:

$$\mathcal{G}(\Phi) = \sum_{i=1}^{N} \frac{1}{\rho_i(\phi_i)}. \tag{34}$$

Clearly, the goal of the power control algorithm is setting the frequency vector Φ to maximize $\mathcal{F}(\Phi)$ subject to power constraints. The same objective is attained via minimization of the denominator function $\mathcal{G}(\Phi)$, with the added benefit of simplifying the objective function expression. Henceforth, $\mathcal{G}(\Phi)$ is selected as the objective function and will be referred to as such in the sequel. $\mathcal{G}(\phi)$ is a nonlinear mapping of application throughputs, and its analytic form cannot be ascertained since throughput analytic form is unknown as established in Chapter 4.

Denote by TDP the thermal design point of the chip, by $B = TDP - P_c$ the chip power budget allocated to cores, and by $[\phi_{min}, \phi_{max}]$ the permissible frequency range available at each core, where ϕ_{min} and ϕ_{max} are the minimum and maximum frequency values, respectively. The chip power control problem is formally defined as:

$$\begin{aligned} \underset{\Phi}{\text{minimize}} \quad & \mathcal{G}(\Phi) \\ \text{subject to} \quad & \mathcal{P}(\Phi) = B, \\ & \phi_{min} \leq \phi_i \leq \phi_{max} \end{aligned} \tag{35}$$

Solving problem (35) directly requires handling the nonlinear constraint set $\mathcal{P}(\Phi) = B$. Alternatively, consider a scheme where the core frequency vector is indirectly calculated using the core power controllers discussed in chapter 3 to track vector of core power setpoints $S = [s_1 \ s_2 \ \dots \ s_N]^T$. If the condition $P_i(\phi_i) = s_i$ holds, an equivalent formulation of the the power control problem is to find the power setpoint vector S that minimizes the objective function subject to power constraints. Compared to problem (35), the alternative formulation has strictly linear constraints, which reduces the computational complexity of the power control algorithm. Formally, the problem is expressed as:

$$\begin{aligned} \underset{S}{\text{minimize}} \quad & \mathcal{G}(S), \\ \text{subject to} \quad & \sum_{i=1}^{N} s_i = B, \\ & \ell_i \leq s_i \leq u_i \end{aligned} \tag{36}$$

where $\ell_i = P_i(\phi_{min})$ and $u_i = P_i(\phi_{max})$ are the power bounds that determine the feasible setpoint range $[\ell_i, u_i]$ at each core. The algorithm solving problem (36) will be referred to as the *master algorithm*, and is detailed in the next section.

5.3 The Master Algorithm

Consider the iterative setpoint update equation given by:

$$S_{k+1} = S_k - \lambda_k X_k, \tag{37}$$

where k is an iteration counter, $X_k \in R^N$ is called the *search direction* vector, and λ_k is a scalar called the step size. In the context of unconstrained convex optimization, setting X_k to the gradient of the objective function $\nabla\mathcal{G}(S)$ yields the classic gradient descent method, which produces a sequence S_k that converges to the optimum [59]. However, in the case of constrained optimization problems, the direction vector X_k must ensure that the sequence S_k converges to the optimum while satisfying constraints at every iteration. Within the master algorithm, the direction vector is calculated using *gradient projection* [59]. Fig. 24 presents a high level view of the solution methodology. Details of the individual components are given in the sequel.

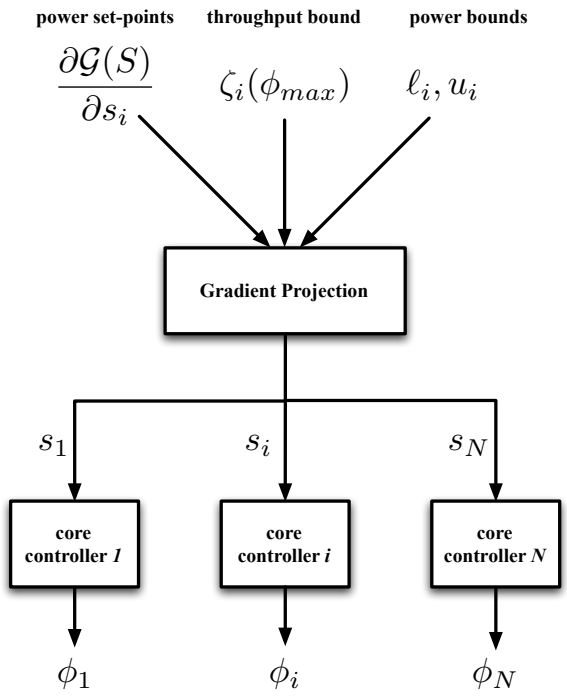


Figure 24: High-level view of the solution methodology

5.3.1 Direction Vector Calculation

The iteration counter k is dropped from the exposition in this section to simplify notation. Without loss of generality, assume that the setpoint vector is defined as follows:

$$S = \begin{cases} \ell_j : & j \in [1, L] \\ u_j : & j \in [L+1, L+U] \\ \ell_j < s_j < u_j : & j \in [L+U+1, N]. \end{cases} \tag{38}$$

Setpoints $s_j : j \in [1, L]$ cannot be *decreased* since they are already at their lower bound ℓ_j. Similarly, setpoints $s_j : j \in [L+1, L+U]$ cannot be *increased* since they are already at the upper bound u_j. These constraints on X are expressed using the following set of functions $f_j(X) \leq 0\ ,\ j \in [1, L+U]$

$$f_j(X) = \begin{cases} x_j : & j \in [1, L] \\ -x_j : & j \in [L+1, L+U] \end{cases} \tag{39}$$

where $f_j : R^N \rightarrow R$. The remaining setpoints $s_j \in [L+U+1, N]$ have no constraints on their update direction. To ensure that the power budget constraint is satisfied, it is assumed that the algorithm is initialized with feasible setpoints; i.e. satisfying $\sum_{i=1}^{N} s_i = B$, and that the feasibility is maintained throughout the subsequent iterations via the following equality constraint:

$$h(X) = \sum_{i=0}^{N} x_i = 0. \tag{40}$$

The method of gradient projection calculates the direction vector by projecting the objective function gradient $\nabla \mathcal{G}(S)$ into the feasible set represented by $h(X) = 0$ and $f_j(X) \leq 0$, $j \in [1, L+U]$. Define the distance function $f_0(X)$ as:

$$f_0(X) = \frac{1}{2} \| X - \nabla \mathcal{G}(S) \|^2. \tag{41}$$

The direction-vector calculation problem can be posed as the following quadratic program:

$$\begin{aligned} \underset{X}{\text{minimize}} \quad & f_0(X), \\ \text{subject to} \quad & h(X) = 0, \\ & f_j(X) \leq 0 \ , \ \forall j \in [1, L+U] \end{aligned} \tag{42}$$

The objective $f_0(X)$ and the inequality constraint functions $f_j(X)$, $j \in [1, L+U]$ are convex, and the equality constraint function $h(X)$ is affine. Therefore, the solution of the quadratic program above must satisfy the Karush-Kuhn-Tucker (KKT) optimality conditions (KKT) [59]. Define the Lagrangian:

$$L(X, \nu, \mu) = f_0(X) + \sum_{j=1}^{L+U} \nu_j \ f_j(X) + \mu \ h(X), \tag{43}$$

where ν_j is the *Lagrange multiplier* associated with the j-th inequality constraint $f_j(X)$, and μ is the multiplier associated with the equality constraint $h(X)$. For an optimal X, the Lagrangian gradient must equal zero:

$$\nabla f_0(X) + \sum_{j=1}^{L+U} \nu_j \nabla f_j(X) + \mu \nabla h(X) = 0. \tag{44}$$

Moreover, the Lagrange multipliers associated with inequality constraints must be *nonnegative*:

$$\nu_j \geq 0 \ , \ \forall j \in [1, L+U]. \tag{45}$$

The last condition is the *complementary slackness* condition, defined as:

$$\nu_j \ f_j(x) = 0 \ , \ \forall j \in [1, L+U]. \tag{46}$$

Observe that $\nabla f_0(X) = X - \nabla G(S)$, and $\nabla h(X) = \mathbf{1} \in R^N$, where $\mathbf{1} = [1 \ 1 \ \dots \ 1]^T$, and

$$\nabla f_j(X) = \Big[\frac{\partial f_j}{\partial x_1} \ \frac{\partial f_j}{\partial x_2} \ \dots \ \frac{\partial f_j}{\partial x_N} \ \Big]^T \tag{47}$$

where

$$\frac{\partial f_j(X)}{\partial x_j} = \begin{cases} 1 & : \ i = j \ , \ j \in [1, L] \\ -1 & : \ i = j \ , j \in [L+1, L+U] \\ 0 & : \ \text{otherwise} \end{cases} \tag{48}$$

Plugging the above gradient expressions into equation (44) yields the following system of N equations:

$$\begin{cases} x_j - \frac{\partial \mathcal{G}(S)}{\partial s_j} + \nu_j + \mu = 0 & : \ j \in [1, L] \\ x_j - \frac{\partial \mathcal{G}(S)}{\partial s_j} - \nu_j + \mu = 0 & : \ j \in [L+1, L+U] \\ x_j - \frac{\partial \mathcal{G}(S)}{\partial s_j} + \mu = 0 & : \ j \in [L+U+1, N] \end{cases} \tag{49}$$

which can be solved for ν_j :

$$\nu_j = \begin{cases} -x_j + \frac{\partial \mathcal{G}(S)}{\partial s_j} - \mu & : \ j \in [1, L] \\ x_j - \frac{\partial \mathcal{G}(S)}{\partial s_j} + \mu & : \ j \in [L+1, L+U] \end{cases} \tag{50}$$

Assume that the objective function derivative terms are always negative $\frac{\partial \mathcal{G}(S)}{\partial s_j} < 0$ (a fact that will be shown in the sequel) and recall that $v_j \geq 0$. To solve for X, consider the first case of equation (50):

1. If $\frac{\partial \mathcal{G}(S)}{\partial s_j} - \mu < 0$, then $x_j < 0$, to maintain the nonnegativity of v_j. By the complementary slackness condition , $v_j = 0$ and $x_j = \frac{\partial \mathcal{G}(S)}{\partial s_j} - \mu$.

2. If $\frac{\partial \mathcal{G}(S)}{\partial s_j} - \mu \geq 0$, $x_j = 0$ by the complementary slackness condition.

Consider now the second case of equation (50):

1. If $-\frac{\partial \mathcal{G}(S)}{\partial s_j} + \mu < 0$: then $x_j > 0$, to maintain the nonnegativity of v_j. From the complementary slackness condition, $v_j = 0$ and therefore $x_j = \frac{\partial \mathcal{G}(S)}{\partial s_j} - \mu$.

2. If $-\frac{\partial \mathcal{G}(S)}{\partial s_j} + \mu \geq 0$: $x_j = 0$ by the complementary slackness condition.

The expression for the elements of the direction vector X are therefore given by:

$$X = \begin{cases} \min\{0, \frac{\partial \mathcal{G}(S)}{\partial s_j} - \mu\} & : \ j \in [1, L] \quad (1) \\ \max\{0, \frac{\partial \mathcal{G}(S)}{\partial s_j} - \mu\} & : \ j \in [L+1, L+U] \quad (2) \\ \frac{\partial \mathcal{G}(S)}{\partial s_j} - \mu & : \ j \in [L+U+1, N] \quad (3), \end{cases} \tag{51}$$

where the expressions for the case (3) follow directly from the third case in equation (49). The remaining unknown μ is found by solving the equation:

$$h(\mu) = \sum_{i=1}^{N} x_i(\mu) = 0. \tag{52}$$

Equation (52) is a piecewise linear equation an instance of which is illustrated in Fig. 25. The *breaking points* represent the points at which the expression of direction-vector element x_i belonging to case (1) or (2) in equation (51) is switched; $0 \rightarrow \frac{\partial \mathcal{G}(S)}{\partial s_j} - \mu$ or vice versa. In the absence of breaking points; e.g. if all x_i's belong to case (3), the *root* of equation 52 is simply $\mu = \frac{1}{N}\sum_{i=1}^{N} \frac{\partial \mathcal{G}(s)}{\partial s_i}$. Otherwise, an iterative approach can be employed to find an exact or an approximate solution. Equation (52) is reminiscent of the piecewise linear equations solved by the *water-filling* algorithm, employed to solve the problem of partitioning a fixed power budget amongst a pool of wireless transmitters to maximize the overall data rate [59, 60]. The literature includes algorithmic approaches to attain exact or approximate solutions for solving problems with a water-filling structure, which can be adapted to solve equation (52). Appendix C describes an exact algorithm adapted from [60] to find the root of equation (52).

5.3.2 $\mathcal{G}(S)$ Derivative and Bound Estimation

The derivative of the objective function $\mathcal{G}(S)$ with respect to each power setpoint is expressed as:

$$\frac{\partial \mathcal{G}(S)}{\partial s_i} = \frac{-\zeta_i(\phi_{max})\zeta_i'(s_i)}{\zeta_i(s_i)^2} \tag{53}$$

where $\zeta_i(\phi_{max})$ is an estimate of the maximum throughput achieved at i-the core, $\zeta'(s_i)$ is the power-throughput derivative, and $\zeta_i(s_i)$ is the measured throughput. The power-throughput derivative is obtained via the chain rule as:

$$\zeta_i'(s_i) = \frac{\zeta_i'(\phi_i)}{P_i'(\phi_i)}, \tag{54}$$

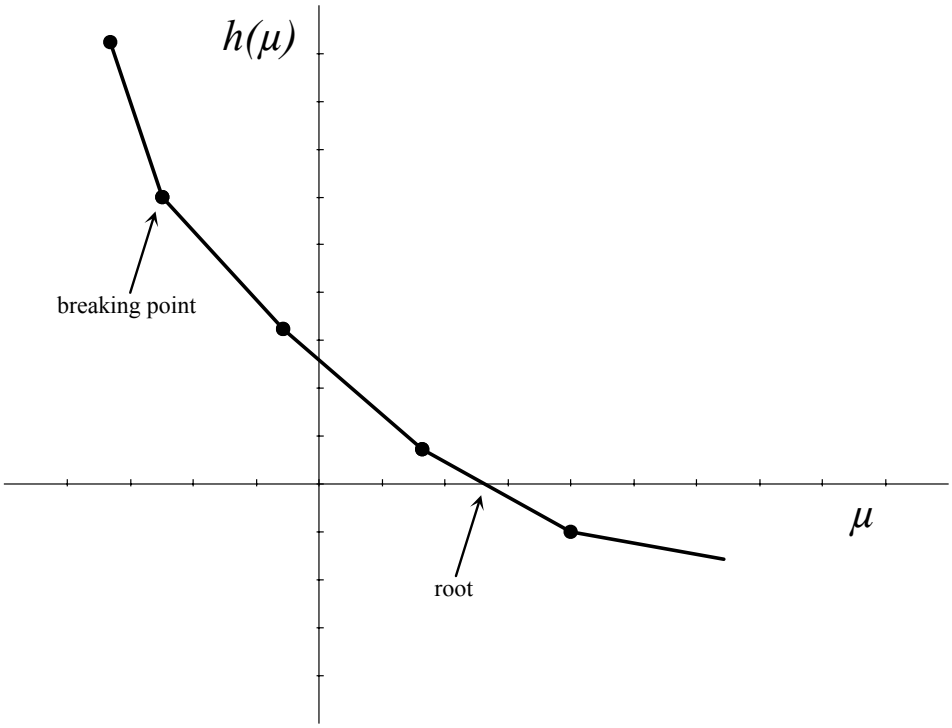


Figure 25: Solution of the piecewise linear equation $h(\mu) = 0$

where $\zeta_i'(\phi_i)$ and $P_i'(\phi_i)$ are the derivatives of the frequency-throughput and frequency-power, respectively. The throughput bound $\zeta_i(\phi_{max})$ is estimated at runtime using linear interpolation using the IPA derivative:

$$\zeta_i(\phi_{max}) = \zeta_i(\phi_i[k]) + (\phi_{max} - \phi_i[k])\zeta_i'(\phi_i), \tag{55}$$

and the power bound estimates are obtained similarly using the power derivative:

$$u_i = s_i + (\phi_{max} - \phi_i)P_i'(\phi_i) \tag{56}$$

$$\ell_i = s_i + (\phi_i - \phi_{min})P_i'(\phi_i) \tag{57}$$

5.4 Simulation Results

The proposed chip-power control algorithm is evaluated using *Zesto* [53]; a detailed cycle-level microprocessor that is integrated with the McPAT [54] tool for microarchitectural power modeling. As illustrated in Fig. 26, the simulated multicore processor configuration comprises four homogenous cores equipped with private L1 and L2 cache memories that are connected via a mesh network. The parameters of the core and cache configurations are listed in Table 5.

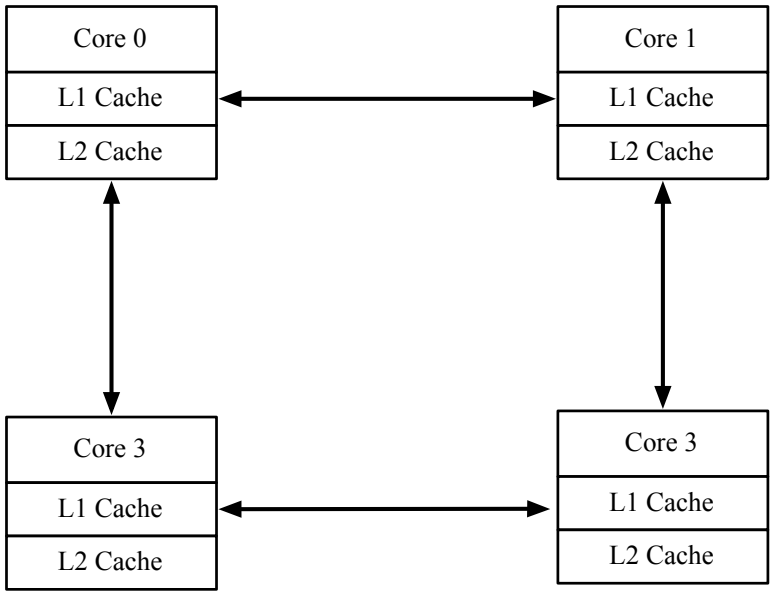


Figure 26: Simulated Multicore Configuration for Chip power control.

Table 5: Simulated processor configuration for throughput regulation

Parameter	Configuration Value
Instruction set Architecture	x86 IA32
Reorder Buffer Size	128 entries
Execution Width	4
L1 Cache	4-way 32B-line 64KB, LRU
Shared L2 Cache	16-way 32B-line 128 KB
Dram Latency	70 ns

The setting simulates multiprogrammed workloads drawn from the SPEC2006 and PARSEC benchmark suites, and the benchmark mixes are listed in Table 6. The simulation runs start with fast forwarding 1×10^9 instruction to initialize the processor

and cache state, and is followed by detailed architectural and power simulation for 250×10^6 instructions.

Table 6: SPEC2006 and PARSEC benchmark mixes

Code	Workload Mix
1	calculix , sjeng , astar-bigl , mcf
2	gromacs, bzip2-comb , bzip2-lib,soplex
3	perl-diff , omnetpp , hmmer-retro , perl-chk
4	bzip2-chk , sphinx3 , spolex , lbm
5	gcc , h264 , libq , mcf
6	h264, h264 , h264 , h264
7	h264 , hmmer , bzip2-comb, sphinx3
8	h264 , mcf , bzip2
9	h264, perl-chk, astar , lbm
10	libq, lbm , bzip2, soplex
11	libq, sjeng , gcc , soplex
12	omnetpp, xalan, hmmer, lbm
13	perl-diff, deal, gcc, bzip2
14	perl-diff, gromacs , xalan, astar
15	perl-diff, omnetpp, hmmer, perl-chk

The discrete voltage-frequency settings available to each core are listed in Table 7, and are used to empirically estimate the parameters of the affine voltage-frequency relationship $V(\phi) = 0.4 + 0.2 \times 10^{-3}\ \phi$, where ϕ is in MHz.

Setting	Frequency (MHz)	Voltage (V)
1	1500	0.70
2	1800	0.76
3	2100	0.82
4	2400	0.88
5	2700	0.94
6	3000	1.0

Table 7: Supported Core Voltage-Frequency Settings

The frequency values calculated by the core power controllers ϕ_i are continuous and may fall outside the discrete supported levels. Therefore, a sigma-delta frequency modulator is used to approximate ϕ_i by generating a sequence of 10 supported frequency values ϕ_i; an approach that has been employed by several proposals in the

literature [46, 24, 14]. Each frequency value within the sequence is held active for a period $T_{\Sigma\Delta} = 25\mu s$, which is inline with the technology trends that project DVFS switching periods as low as $250ns$ [14]. The core controllers and the master algorithm are invoked at the same rate with a period of $T_c = 250\mu s$.

5.4.1 Baseline Algorithm

The proposed power control algorithm is compared against a state of the art baseline proposed in [14]. The baseline algorithm operates by adjusting the *aggregate* chip frequency using an integral controller to track the power budget, and partitioning the chip frequency quota between cores to optimize the fair speed up metric. Denote by f the chip frequency quota, and by $e(k) = B - \mathcal{P}(\Phi)$ the power tracking error at the k-th control period. The chip frequency quota is adjusted by an integral controller, given by:

$$f[k+1] = f[k] + K_I \; e[k], \tag{58}$$

where K_I is the controller gain that is calculated offline using the processor data sheet to ensure stability. The chip frequency quota $f[k]$ is then divided between cores as follows:

$$\phi_i[k] = \frac{\alpha_i[k]}{\sum_{j=1}^{N} \alpha_j[k]} \; f[k], \tag{59}$$

where $\alpha_i[k]$ is the ratio of *normalized* throughput to power at each core, defined as:

$$\alpha_i[k] = \frac{\zeta_i(\phi_i[k])}{\langle \zeta_i(\phi_{max}) \rangle} \frac{1}{P_i(\phi_i[k])}. \tag{60}$$

The $\langle \zeta_i(\phi_m ax) \rangle$ is the average application throughput at ϕ_{max} acquired via offline profiling. The fixed controller gain K_I is likely to degrade the settling time since the plant (chip power as a function of the frequency budget f) is time varying and

will invariably deviate from the offline-assumed plant. Moreover, the scheme incurs nontrivial profiling overhead since the it requires estimating the offline estimation of peak through$\langle\zeta_i(\phi_{max})\rangle$ for every application. Finally, there is no guarantee that the core frequencies calculated by Equation (59) will satisfy the constraint $\phi_{min} \leq \phi_i \leq \phi_{max}$, since the α_i's are time varying, a fact that is formally shown in Appendix B.

The baseline is simulated with a range of controller gain values $K_I = [5, 6, 7, 8, 10, 20, 30, 40, 50]$ to illustrate the effect of K_I of the tracking and chip performance of the baseline algorithm.

5.4.2 Power Tracking Analysis

The tracking mean-squared error ε is used to compare the tracking quality of the evaluated techniques. Denote by $e[k] = B - \mathcal{P}[k]$ the tracking error at the k-th control period, and by L the number of control periods in a simulation experiment. The tracking mean-squared error is defined as:

$$\varepsilon = \frac{1}{L}\sum_{k=1}^{L} e^2(k) \tag{61}$$

The workload mixes 2 and 6 are selected as a case study illustrating the tracking accuracy of the controllers under diverse workload scenarios. Figures 27 and 28 illustrate the *uncontrolled* power envelope of the workload mixes, where all the cores are running at the maximum frequency ϕ_{max}. The power values (y-axis) are normalized by the chip power budget B. The uncontrolled power envelope of workload 2 is fairly stable, and slightly exceeds the power budget B. On the other hand, the power envelope of workload 6 violates the budget by a greater extent and is more oscillatory.

The controlled power envelopes are shown in Fig. 29 and 30 for workload mix 2 and 6, respectively. Plotted in the figures are power envelopes of the proposed adaptive algorithm, as well as the baseline algorithm with gain values $K_I = 8$ and $K_I = 20$, which yielded the lowest mean-squared error ε for workloads 2 and 6, respectively. Fig.

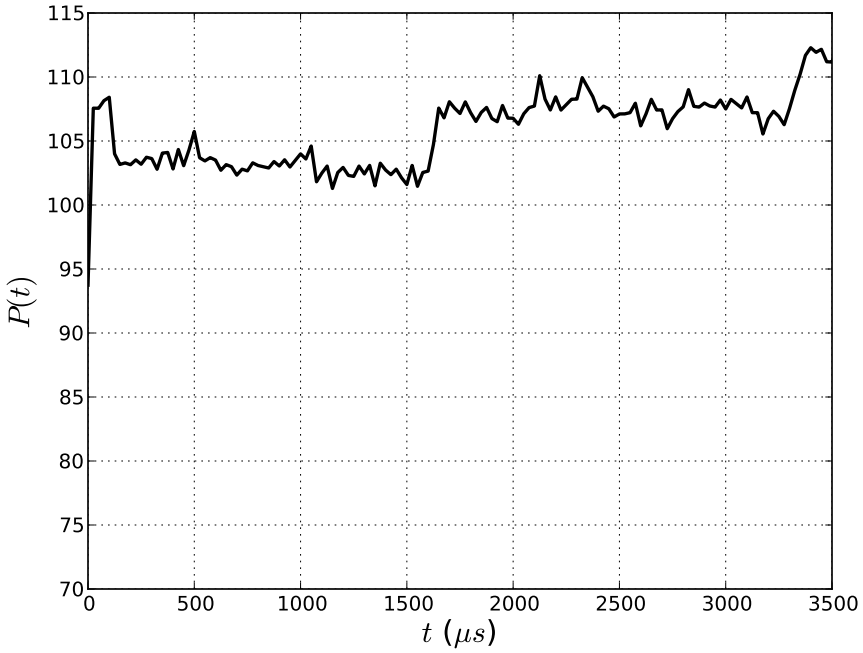


Figure 27: Uncontrolled Chip power envelope for workload mix 2

29 shows that $K_I = 8$ represents the better fit for this workload mix, since it yields the fastest settling time and least oscillations amongst the fixed gain values, and is comparable to the settling time of the proposed adaptive algorithm, In contrast, Fig. 30 shows that $K_I = 20$ is a better fit for workload mix 6, since $K_I = 8$ causes high oscillations in the power envelope. The proposed algorithm yielded faster settling time and less oscillations compared to both instances of the baseline algorithm. The past analysis illustrates the advantage of an *adaptive* scheme that sets the control parameters based on the active workload to ensure rapid settling times.

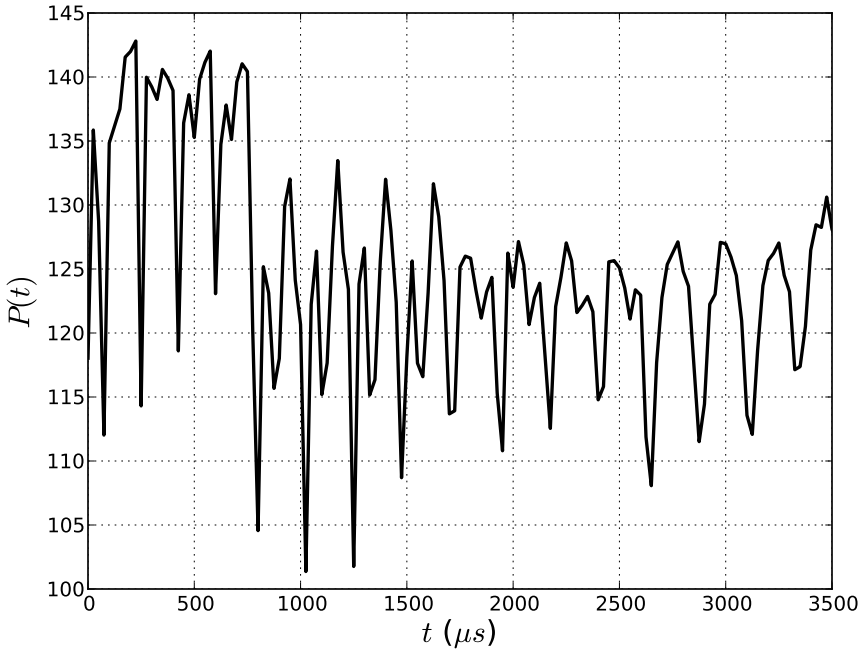


Figure 28: Uncontrolled Chip power envelope for workload mix 6

Figures 31-34 present a detailed view of the operation of the proposed adaptive algorithm for workload mix 2. Fig. 31 shows the absolute value of the objective function derivatives $|\frac{\partial \mathcal{G}(S)}{\partial s_i}|$ evaluated at each core. Observe that in terms of the derivative magnitudes, the core 4 is the largest, followed by core 3, then cores 2 and 1, which have comparable magnitudes. The core power setpoints are plotted in Fig. 32, and the corresponding frequency values calculated by the core controllers are plotted in Fig. 33. The frequency assignment, plotted in Fig. 33 , is more reflective of the objective function gradients, where the higher the magnitude of the derivative, the higher the frequency assignment ϕ_i calculated by the core controllers. Finally, Fig. 34 plots the tracking error at each core, which is rapidly reduced and maintained around 7%.

Fig. 35 is a scatter plot that compares the tracking mean-squared error ε of the proposed adaptive algorithm to the gain-swept baseline algorithm across all benchmarks. The figure demonstrates that the proposed adaptive algorithm consistently

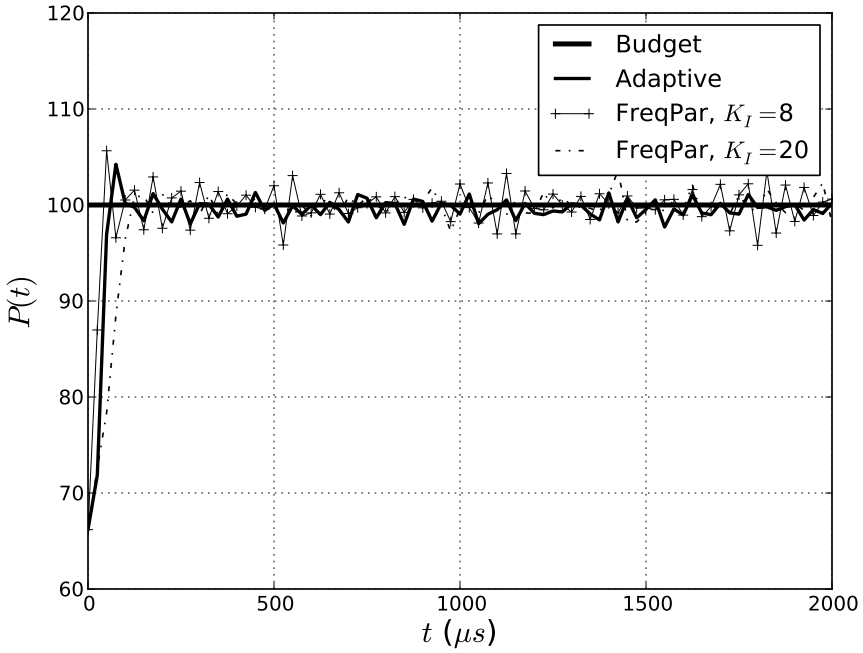


Figure 29: Controlled Chip power envelope for workload mix 2

delivers the highest tracking accuracy compared to fixed-gain baseline algorithm of [14]

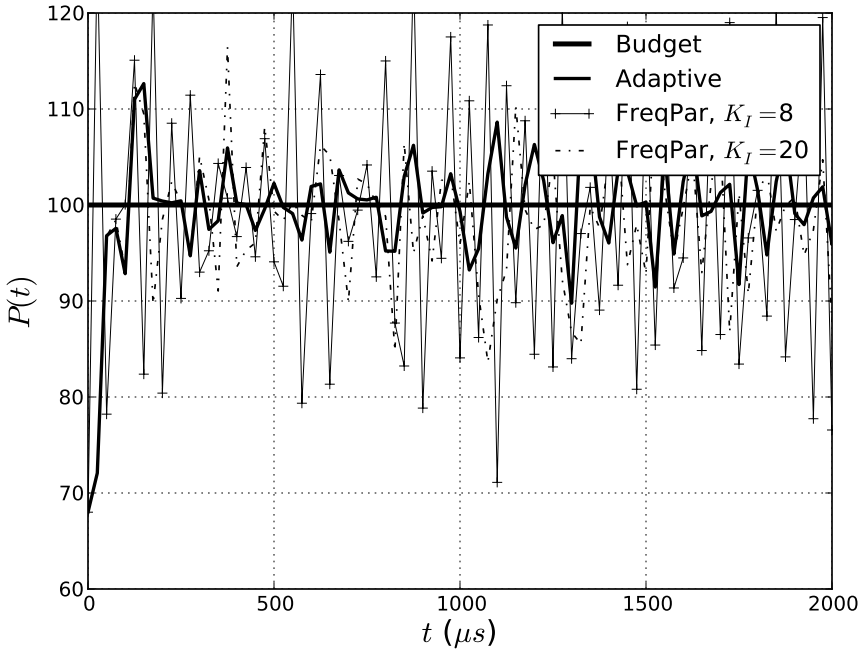


Figure 30: Controlled Chip power envelope for workload mix 2

5.4.3 Performance Optimization Analysis

To illustrate how the proposed algorithm optimizes the fair speed up metric, consider workload mixes 2 and 6 whose intrinsic application throughputs are plotted in Fig. 36 and 37, respectively. The applications in workload mix 2 vary in their average throughput level, but in general, their throughput is stable and does not experience major oscillations. On the other hand, workload mix 6 comprises four instances of the same benchmark *h264*, and they exhibit similar throughput behavior, which is highly oscillatory.

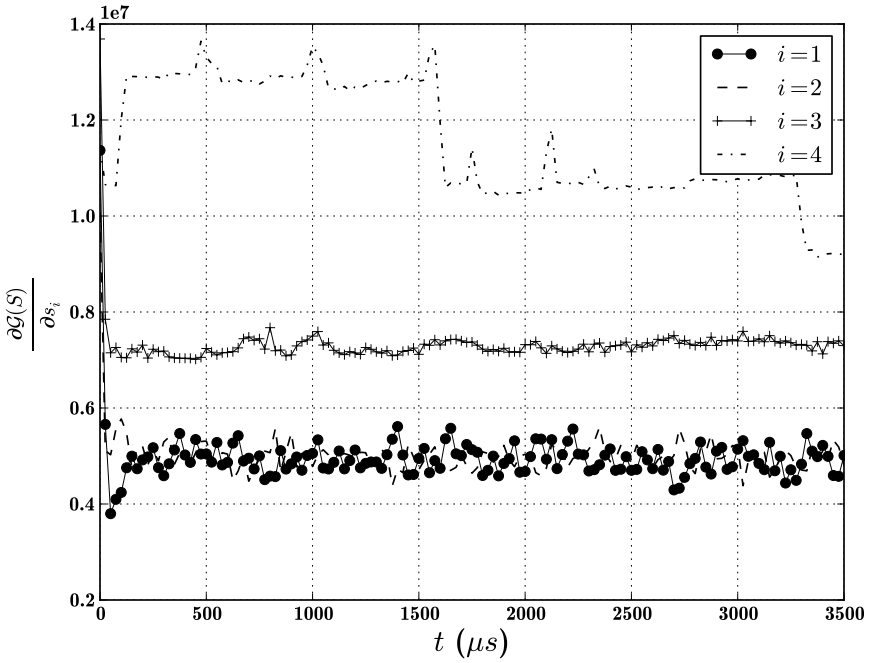


Figure 31: Derivatives $|\frac{\partial \mathcal{G}(S)}{\partial s_i}|$

Fig. 38 show the plots the norm of the objective function $\|\nabla \mathcal{G}(S)\|$, whose magnitude should be minimized to ensure optimality. The figure shows that the proposed algorithm rapidly reduces the objective function norm at the beginning of the run. Subsequently, the norm undergoes some variations due to workload variations.

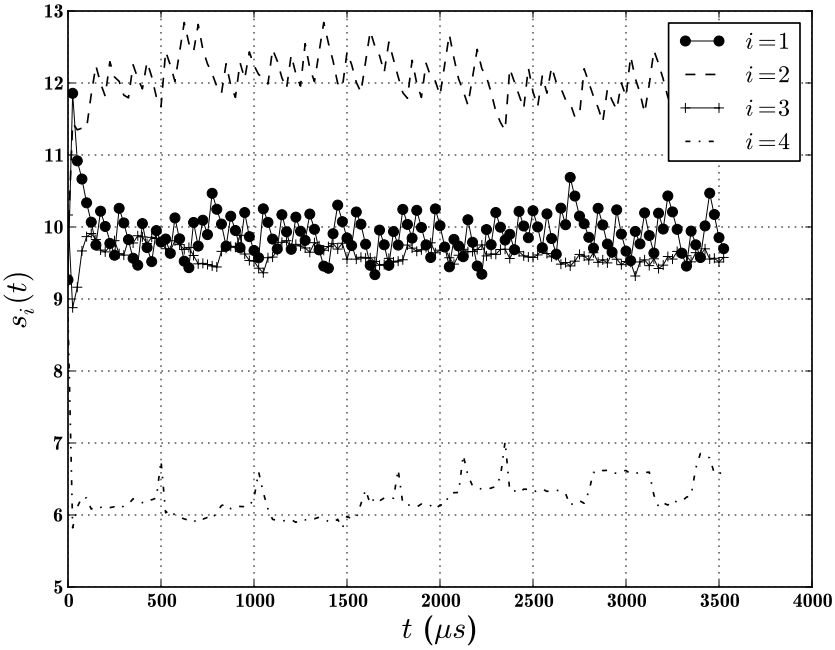


Figure 32: Setpoints s_i

Fig. 39 is a scatter plot comparing the fair speedup values $\mathcal{F}$ yielded by the proposed algorithm to those of the swept-gain baseline algorithm across all workload mixes. Clearly, the proposed algorithm outperforms *FreqPar* by a noticeable margin across all workload mixes.

5.5 Related Work

Intel Montecito; a dual-core Itanium processor, was equipped with a control system comprising power and temperature sensors and a dedicated microcontroller, which successfully controlled power and temperature using chip-wide DVFS [37]. Subsequently, most of the commercial processors came equipped with per-core DVFS settings, which improved the control accuracy. Heuristics were the basis of several chip-power control proposals in the literature [17, 34, 35, 21]. The heuristic approaches included trial-and-error adjustment of core DVFS settings [17, 21], and exhaustive search of the per-core DVFS space using offline-generated power and performance

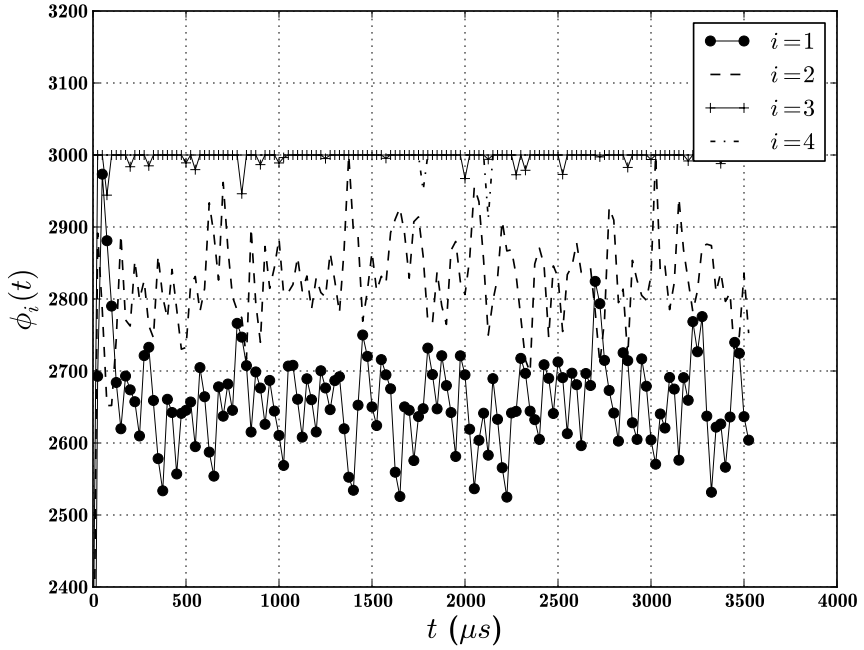


Figure 33: Frequency ϕ_i

predictive models [17]. Later works have shown the drawbacks of heuristic-based approaches, which include prolonged algorithm running times, imprecise control, and limited performance [24, 14].

To increase the robustness to model estimation errors and provide theoretical guarantees on algorithm performance, several chip-power control proposals were based on formal feedback control theory. Mishra *et. al.* [22, 23], proposed a power budgeting scheme where the chip power budget is partitioned between voltage islands to maximize chip performance. The power setpoints assigned to each voltage island are tracked by adjusting the local DVFS settings using PID control. Stability of the local PID controllers is not guaranteed since they are designed using average offline analysis. Moreover, since this approach does not account for the limited achievable power ranges available at each voltage-island, which arise due to the limited DVFS settings, the approach may calculate infeasible power partitions that undermine the preciseness of power tracking and the delivered chip performance.

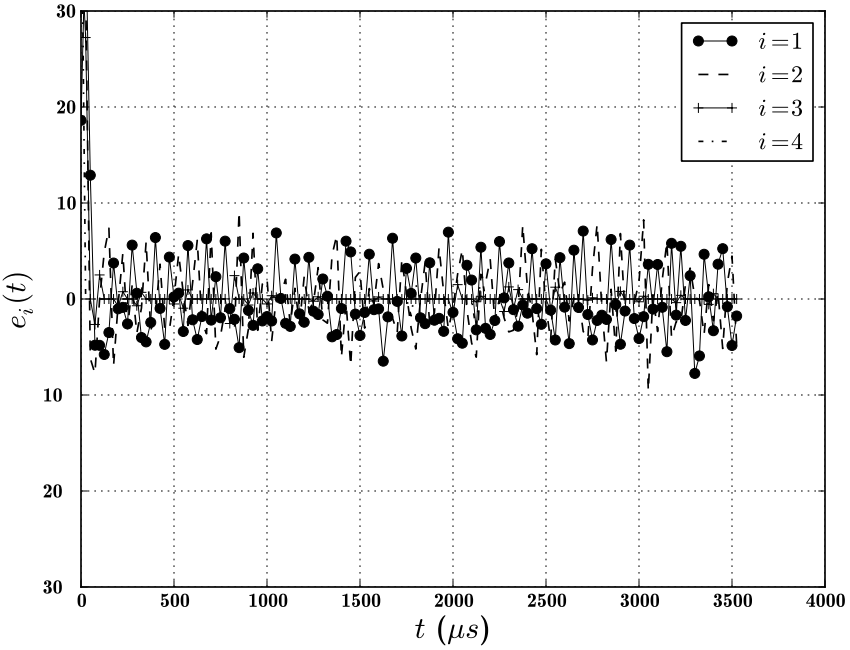


Figure 34: Tracking Error e_i

Wang *et. al.* [24, 36] proposed a centralized scheme based on model-predictive control (MPC) [24]. The relationship between core DVFS settings and chip power was modeled as a linear memoryless dynamical system, whose parameters were estimated at runtime using online observations. The authors demonstrated successful power and temperature control on a quad-core Intel Xeon processor. However, the scheme yielded high computational complexity and algorithm running time due to its centralized nature and the online model estimator which requires matrix inversion [14].

Ma *et. al.* [14], proposed a power control scheme for many-core processors running single- and multi-threaded workloads. The scheme uses an integral controller that adjusts the chip *frequency quota*, defined as the aggregate frequency of all the cores, to ensure the chip power envelope tracks the budget. The gain of the integral controller is calculated offline to guarantee stability by assuming worst-case plant conditions. The chip frequency quota calculated by the integral controller is partitioned

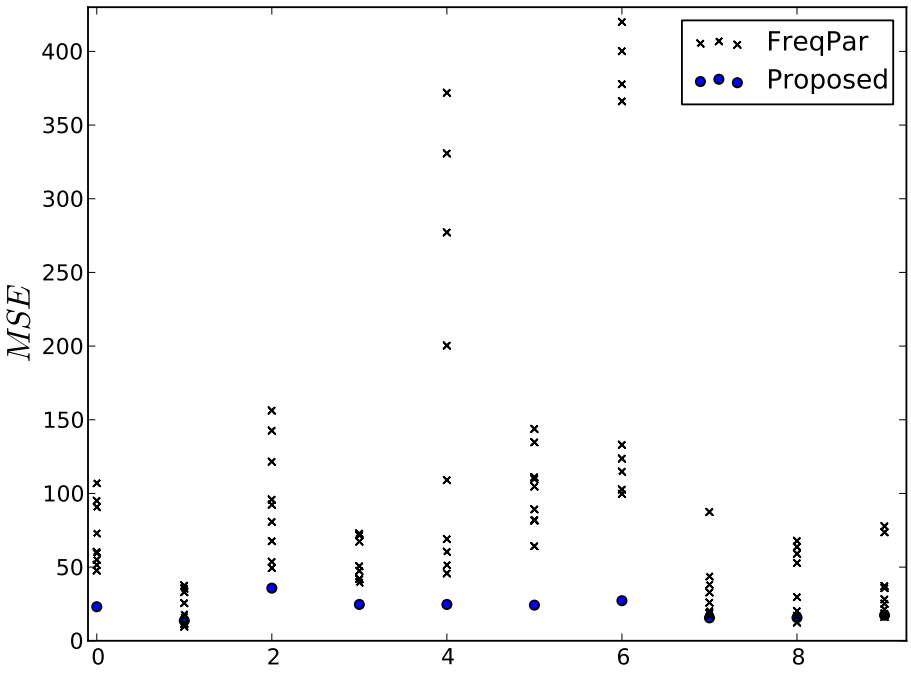


Figure 35: Tracking mean-squared error for all workload mixes

between the cores to maximize *fair* performance. The frequency-quota partitioning algorithm used by the proposal does not account for the limited range of DVFS settings available at each core. Thus, it may calculate infeasible DVFS settings that negatively impact chip power tracking and performance. Moreover, the reliance of the scheme on offline analysis, both in the design of the power controller and in estimating application performance necessary for fair performance maximization, makes the scheme less adaptive to the rapid variations in power and performance behaviors at runtime. In contrast, the chip power control scheme proposed by this dissertation achieves adaptation by characterizing the power and performance behaviors online using derivative estimation. Moreover, unlike the aforementioned chip power control schemes, it accounts for the constraints imposed by limited core DVFS settings.

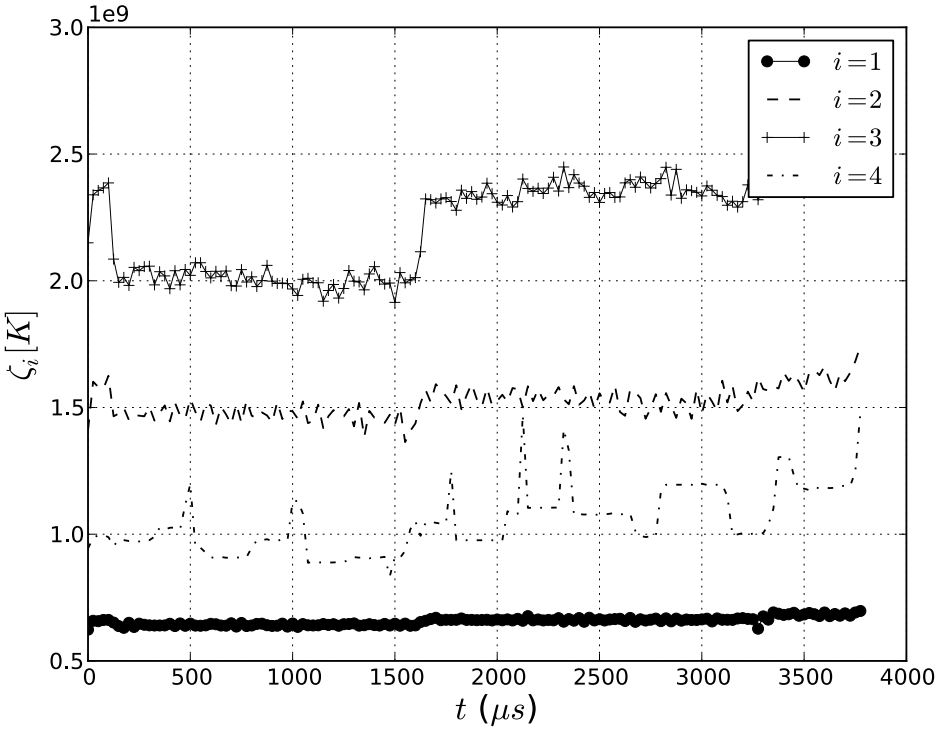


Figure 36: Throughput at ϕ_{max} for workload mix 2

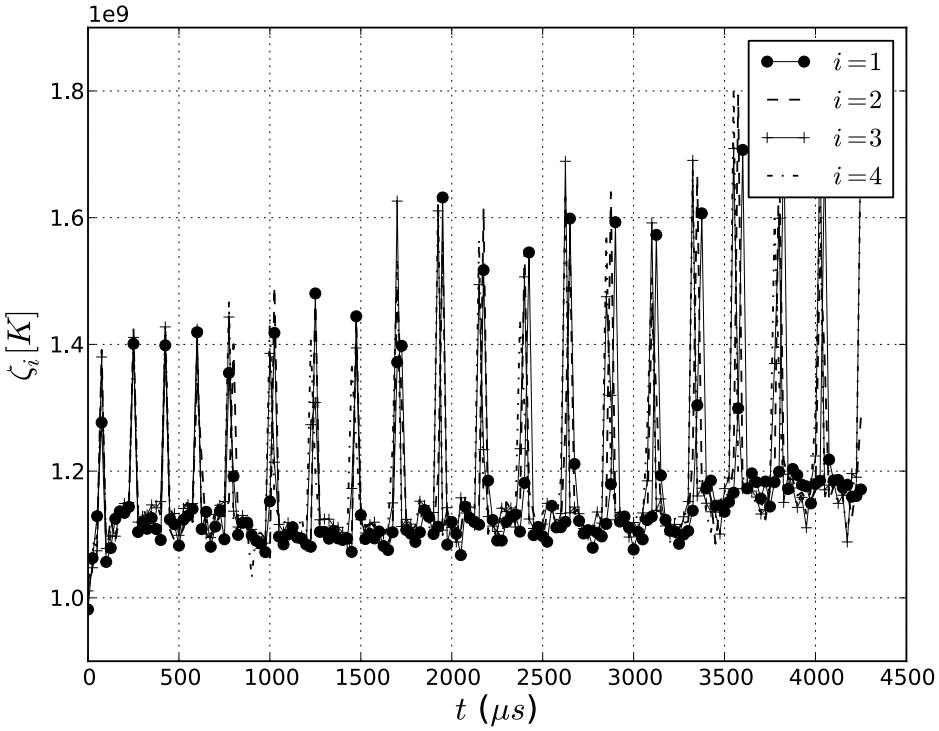
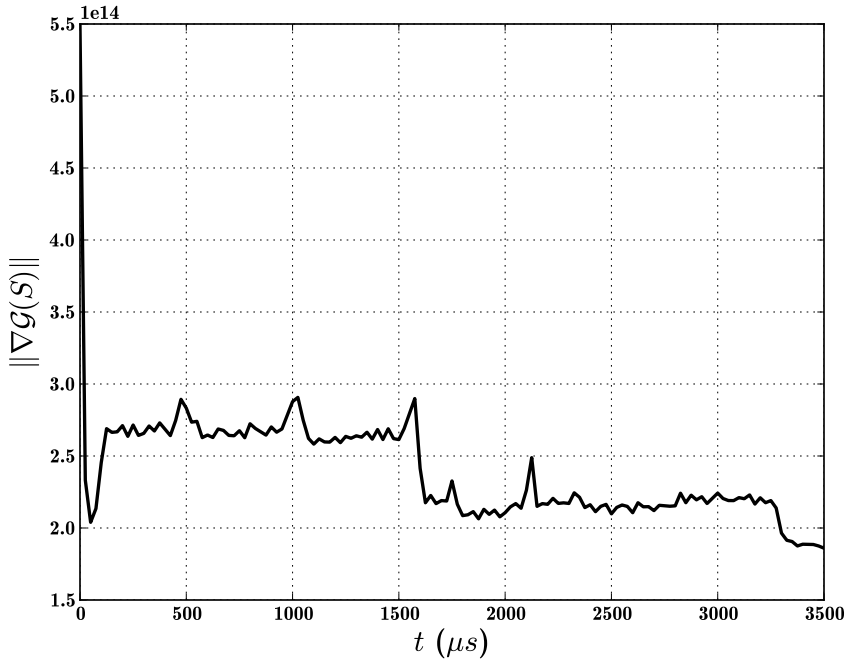
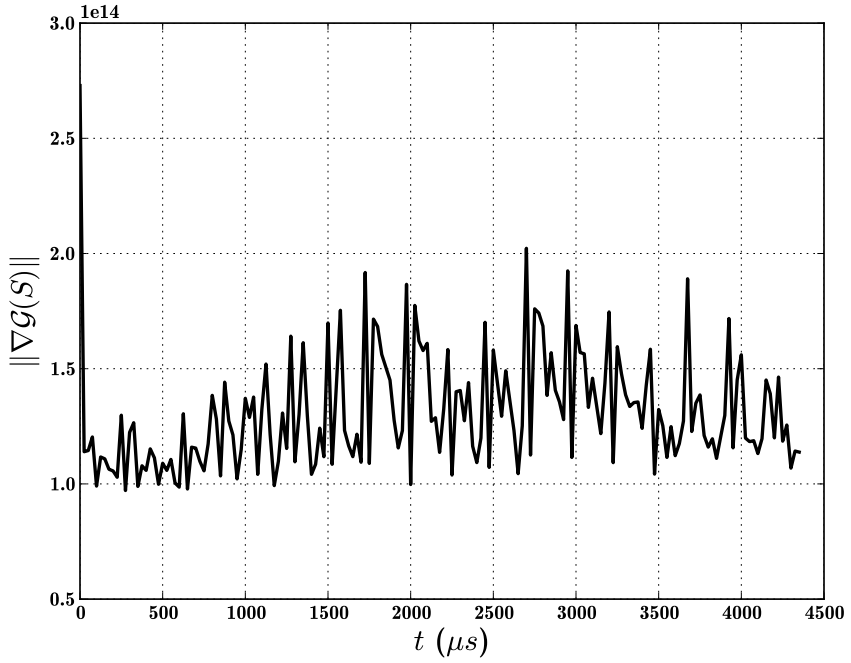


Figure 37: Throughput at ϕ_{max} for workload mix 6



(a) workload mix 2



(b) *workload mix* 6

Figure 38: Objective Function Norm $\nabla\mathcal{G}(S)$ for workload mixes 2 and 6

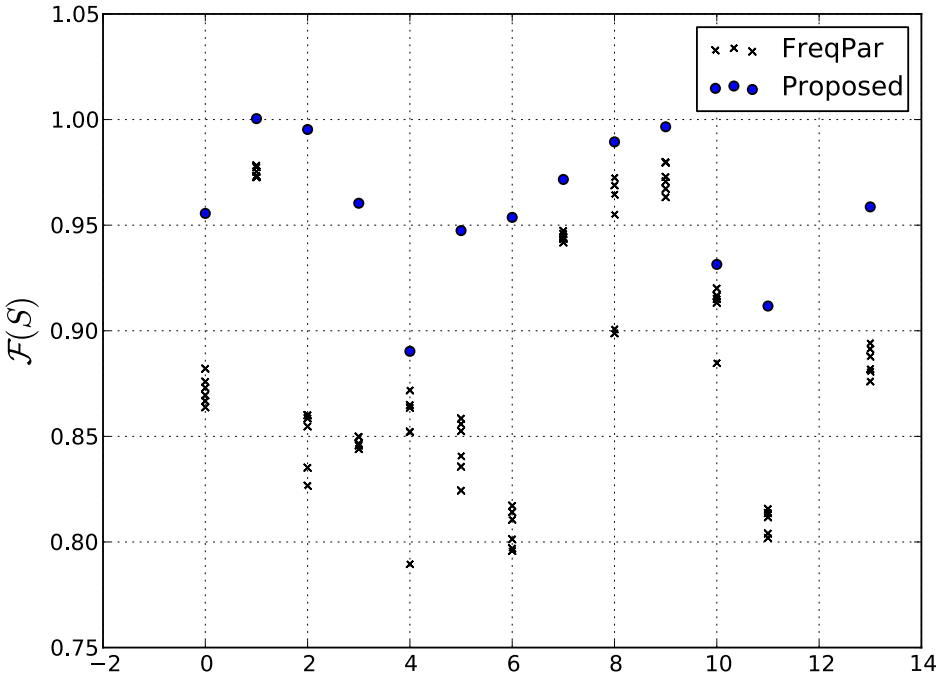


Figure 39: Fair speed $\mathcal{F}$ for all workload mixes

CHAPTER VI

CACHE ENERGY MINIMIZATION

6.1 Overview

Improving the energy efficiency of cache memories using adaptive mechanisms has been the goal of numerous proposals [27, 61, 38, 39], which attempt to minimize unnecessary energy costs during cache operation. An example of such energy costs arises at the interval demarcated by the last access to a cache line and it's eviction. During this interval, the values held by the lines are no longer needed by the program, yet they remain in the cache and consume leakage energy. Cache decay [27] is a mechanism that identifies dead cache lines and minimizes their leakage energy by switching them off. It predicts a cache line to be dead if it remains *idle* for a duration longer than a threshold termed the *decay interval*. Cache decay can result in significant leakage-energy savings. However, it might erroneously turn off active cache lines and increase the miss rate. The misses *induced* by cache decay incur leakage-energy cost at the processor, which remains idle pending miss service. The decay interval is therefore set to balance the tradeoff between energy savings at the cache and the energy costs incurred at the processor. Figure 40 illustrates the relationship between energy and the value of the decay interval (in cycles) , which is denoted as c. The figure is generated for the program *vortex* from the SPEC2000 benchmark suit, using a simulation setup described in section 6.3.

Cache decay is implemented in the level-2 cache (L2), since it has higher potential of leakage-energy savings compared to the smaller L1 cache, and as Figure 40 shows, the energy consumption increases in the L2 with c, since a larger c decreases the likelihood of line deactivation. An opposing trend is observed in the core energy,

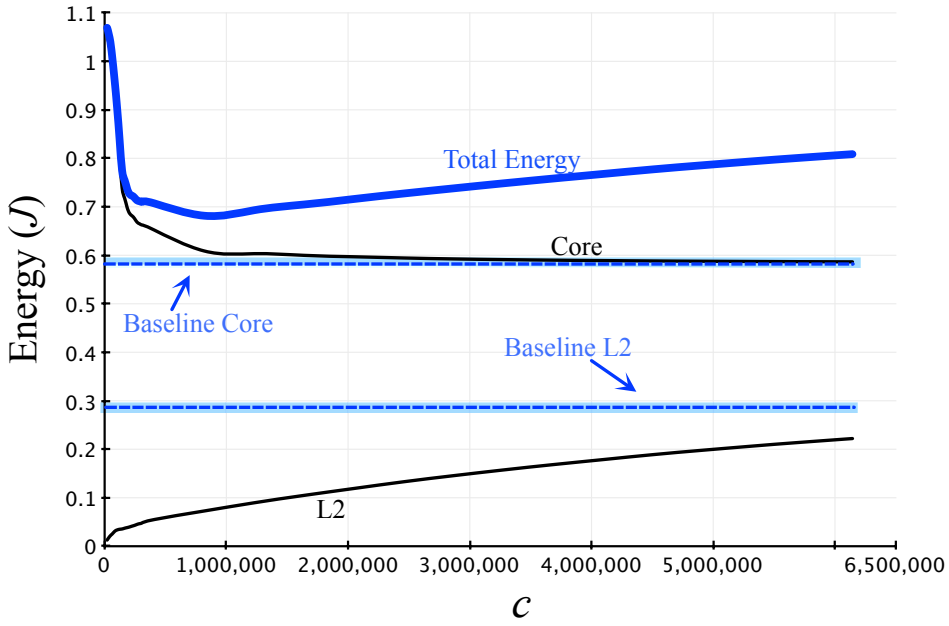


Figure 40: Energy dependence on the decay interval (c).

which decreases with c. This is attributed to the decrease in induced misses and their associated core-energy costs.

The total energy graph (bold line) reveals a convex shape with a unique minimum at $c \approx 800 \times 10^3$ cycles. We refer to the aforementioned value of c as the *fixed minimum*, since each c is fixed throughout execution. Similar experiments were performed over a wider set of SPEC2000 benchmarks. The search range extends from 2-K cycles to 6-M cycles, and the results are summarized in Table 8, which shows significant variation between programs in the energy-saving potential and the decay-interval settings that yields it. This variation motivates the use of schemes that calculate the decay interval at runtime.

6.2 Proposed Solution

Our approach involves posing the decay-interval calculation as an energy-minimization problem, and solving it using a gradient-descent algorithm resembling the Robbins-Monro stochastic approximation algorithm [62]. The total energy is modeled as the sum of two principal components: the cache leakage energy, and the core leakage

Table 8: Decay-interval variation between programs

Name	**Static-Minimum** ($\times 10^3$ cycles)	**Energy Savings** (%)
ammp	2000	11.9
art	20	37.1
bzip	300	26.6
galgel	800	13.5
gcc	2	22.9
twolf	6000	-0.79
vortex	800	21.7
vpr	240	18.8

energy. We derive the sensitivity of both with respect the decay interval, and use them to drive the gradient-descent algorithm.

Denote the leakage energy of the L2 cache by $f_1(c)$, and the number of induced misses by $f_2(c)$, and by P_{idle} the idle leakage power of the core. We model the total energy as a function of the decay interval c as

$$E(c) = f_1(c) + \beta P_{idle} f_2(c), \tag{62}$$

where β is the latency of induced misses. We seek to optimize $E(c)$ using a gradient-descent (GD) algorithm resembling stochastic approximation [62], which is commonly used in sample-path stochastic optimization. If we represent the energy gradient at iteration k as $\frac{dE(c[k])}{dc}$, the GD adaptation is given by

$$c[k+1] = c[k] - \lambda_k \frac{dE(c[k])}{dc}, \tag{63}$$

where $\lambda_k = \frac{\lambda_0}{(k \bmod N)^{0.6}}$ is the step size which controls the speed of convergence, k is an integer representing the iteration number, and mod represents the modulo operation. Using Equation (62), we can express the energy gradient as

$$\frac{dE(c[k])}{dc} = \frac{df_1(c[k])}{dc} + P_{idle}\beta \frac{df_2(c[k])}{dc}. \tag{64}$$

The cache leakage energy $f_1(c)$ is a function of line leakage power (P_{line}), and line *on-time* (T_{on}), which is the total time spent by the line in the on-state. T_{on} varies between lines, and is a function of c and the reference arrival time to each cache line. Suppose a cache line was turned off (decayed) N_d times during execution. The *on-time* in this case is $T_{on} = [T_1, T_2, \ldots, T_{Nd}]$, where T_i is the i-th *on*-interval starting at $t = a_i$ and ending with a decay event at $t = b_i$. Hence, the line leakage energy $E_{line}(c)$ is given by

$$E_{line}(c) = \sum_{i=0}^{N_d} P_{line}(a_i - b_i)c, \tag{65}$$

It can be seen that the on-time grows linearly with c (with unit slope); if c increases by some amount, each $T_i \in T_{on}$ increases by the same amount. This indicates that $\frac{dE_{line}}{dc} = P_{line}\ N_d$. Formally, the gradient is given by

$$\frac{dE_{line}(c)}{dc} = \sum_{i=0}^{N_d-1} P_{line}(\frac{d\beta_i}{dc} - \frac{d\zeta_i}{dc}) \tag{66}$$

$$= \sum_{i=0}^{N_d-1} P_{line}. \tag{67}$$

Taking N_d to be the decay actions across all cache lines, we have

$$\frac{df_1(c)}{dc} = N_d\ P_{line}. \tag{68}$$

The function $f_2(c)$ is not differentiable since it is discontinuous. An example of the shape of this function is shown in Figure 41 (b). Instead of the derivative, we adopt the following difference function to represent the *sensitivity* of $f_2(c)$:

$$\Delta f_2(c) = f_2(c) - f_2(c - \Delta c). \tag{69}$$

Figure 41 (a) shows an example arrival sequence of hits to a cache line. By fixing c to the arbitrary value of $c = 4$, it is clear that a single induced miss occurs and

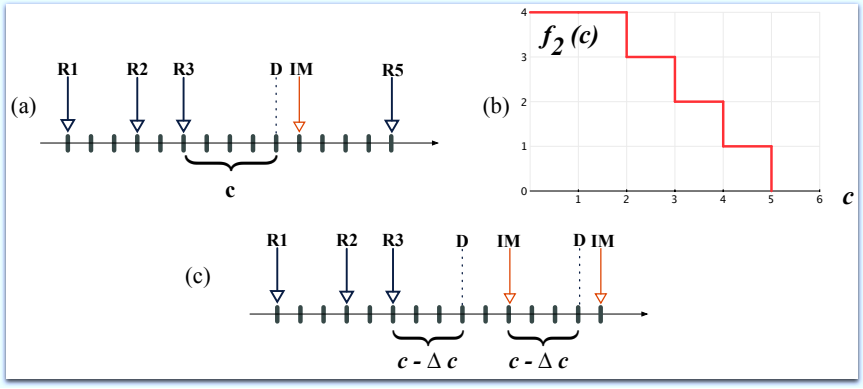


Figure 41: Sensitivity of $f_2(c)$ to changes in c **(a)** hit-arrival Sequence **(b)** $f_2(c)$ plot **(c)** positive perturbation d) negative perturbation

therefore $f_2(c) = 1$. Now, consider the case where c is negatively perturbed (reduced) by $\Delta c = 1$, which causes the last hit, R5, to become an induced miss. This indicates that this reference sequence has one *sensitive hit* when $c = 4$, and in this case $\Delta f_2(c) = 1 - 2 = -1$. In practice, sensitive hits can be determined by checking if the idle-time counter of a line is $\leq \Delta c$ at the time of hit arrival.

Substituting the expressions for $\frac{df_1(c[k])}{dc}$ and $\Delta f_2(c[k])$ into Equation (64), yields

$$\Delta E(c[k]) = LN_d(c[k]) + P_{idle}\beta\Delta f_2(c[k]). \tag{70}$$

The hardware implementation is shown in in Figure 42. The period of the hierarchical counter is $2K$ cycles, and the line idle-time counters are 12-bits long. Hence, as shown in Figure 42(a), each L2 cache-line is augmented with a 12-bit counter, which translates to $\approx 2.1\%$ area overhead. Cache lines are decayed if their idle time exceeds the current value of the decay interval $c[k]$, and N_d is incremented. On the other hand, the value of the idle-time counter is inspected upon each hit, and if it is less than or equal to Δc, Δf_2 is decremented. The sensitivity calculation circuitry is shown in Figure 42(b). The calculation involves addition/subtraction, multiplication, and barrel-shifting to simulate the multiplication by the step size. The step size parameters are set to $\lambda_0 = 0.0005$ and $N = 8$. Based on our experiments, $c[k]$ can

be updated periodically every $T_s = 500 \times 10^6$ cycles, which for a 3 GHz processor, corresponds to ≈ 0.17 millisecond (6 KHz frequency). This makes the gradient approximation amenable for software implementation using a microcontroller similar to the one proposed in [37].

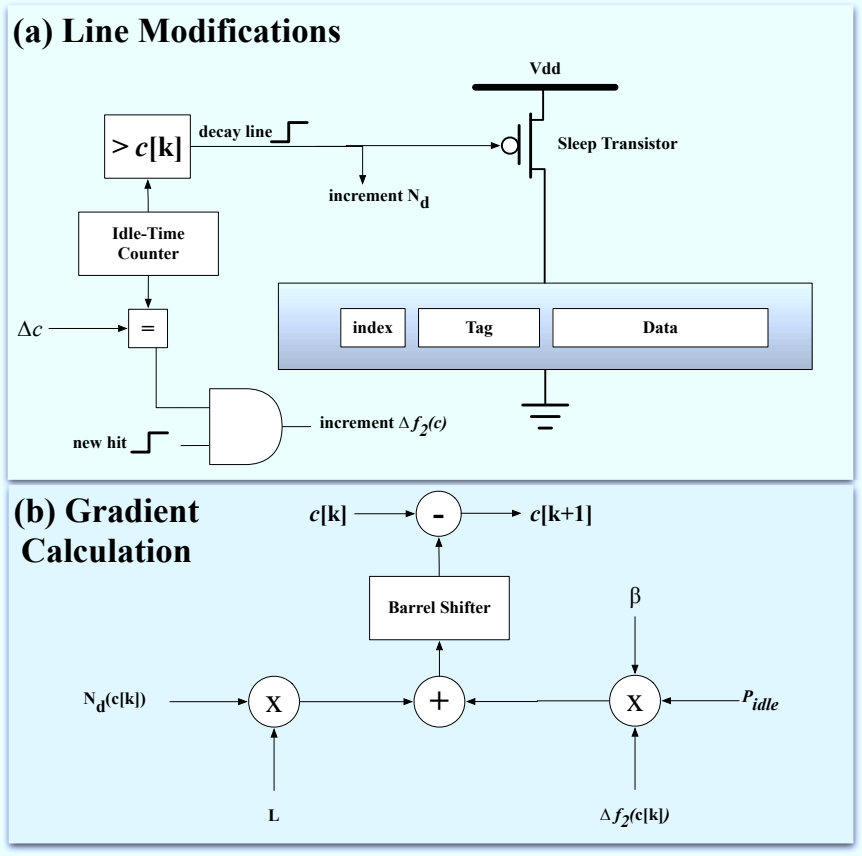


Figure 42: Proposed implementation: (a) cache-line modifications (b) gradient calculation circuitry

6.3 Simulation Results

The proposed optimization is simulated using *HOTLeakage* [63] for 70 nm technology node . The configuration of the simulated machine is summarized in Table 9. Each benchmark is simulated for a period of 250 million instructions, with c ranging between $1 - 2048$ global-counter ticks (2K - 4M clock cycles).

An example of the online adaptation of $c[k]$ using the proposed techniuqe is shown in Figure 43 for *vortex*, whose fixed minimum was shown in Figure 40 to lie between

Table 9: Simulated processor configuration

Core Components and Memory Hierarchy
4-Instructions Issue Width, 80-RUU, 40-LSQ 4 Integer-ALU, 2-FPALU , 1 Integer-MULT, 2 mem-ports 64-KB, 2-W, 64-B, 2-cycle latency LRU L1 (I/D) 1-MB, 8-W, 64-B, 11-cycle latency LRU UL2 250 cycle memory latency, 4-cycle inter-chunk
Technology-related
70nm, 3-Ghz frequency Temperature = 353 °K

approximately 700K - 900-K cycles. The convergence is shown for three different initial decay-interval values $c[0]$, and it is clear that regardless of the initial point the algorithm converges to vicinity of the static minimum indicated by the shaded region.

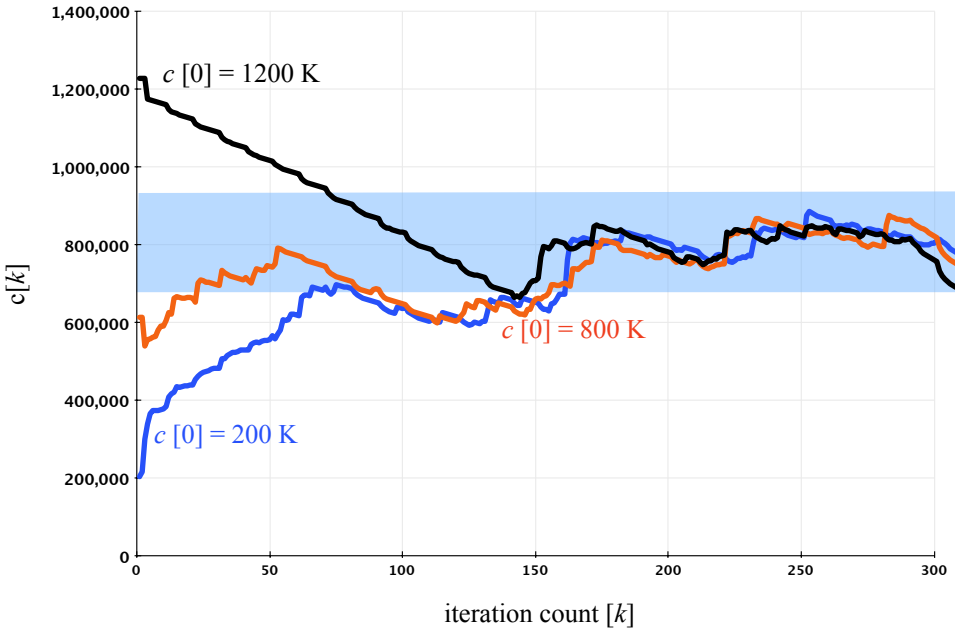


Figure 43: Iterative $c[k]$ update using the gradient-descent algorithm. Shaded region indicates the static minimum

CHAPTER VII

CONCLUSIONS

7.1 Summary

Processor designers adopted manycore architectures to curtail the rising power-consumption levels and maintain performance scaling via parallelism. Despite its success, the manycore paradigm has introduced unprecedented challenges to the design and operation of processors. The diverse applications that execute simultaneously on manycore platforms cause high runtime power and performance variations. Power variations impact the reliability of chips, as well the cost of their cooling and power-delivery systems. Moreover, the variability of performance has an impact on the energy efficiency and performance returns of the manycore processing paradigm. The aforementioned challenges have highlighted the need for runtime power and performance management of manycore processors. However, the design of management algorithms is challenging since power and performance are strongly dependent on the workload, which cannot be determined *apriori* and exhibits wide and rapid runtime variations.

This dissertation seeks to show that sensitivity analysis provides runtime information about the time-varying power and performance behaviors that enables the design of adaptive management algorithms for manycore processors. Towards this goal, the dissertation contributes adaptive algorithms that rely on runtime sensitivity (derivative) estimation to solve the problems of controlling the power and performance of processor cores, maximizing the performance of manycore processors under a fixed power budget, and optimizing the energy consumption of cache memories.

The first contribution is concerned with controlling the power of processor cores, which is an essential component of controlling the power of manycore processors and

maximizing their performance. The design of core-power controllers that guarantee stability and rapid settling is challenging since power is a function of the time-varying workload. The dissertation proposes an integral controller that tracks desired power levels by adjusting core frequency settings. The derivative of the time-varying plant (frequency-power functional relation) is estimated online and is used to adaptively set the controller gain. The proposed adaptive controller is shown formally and via detailed simulation to achieve rapid and robust tracking under diverse workload conditions. In contrast, simulation results show that plant variations can degrade the settling time of fixed-gain controllers designed using offline analysis.

The next contribution is concerned with the problem of regulating application throughputs via adjustment of core frequency settings. Throughput targets are set to ensure quality of service and improve energy efficiency. The design of throughput regulators is however challenging due to the wide range of application behaviors, and the throughput fluctuations introduced by the memory hierarchy and speculative execution employed in out-of-order processing. The dissertation proposes an integral-control algorithm for throughput regulation, where the gain is adaptively set using the sample derivative of the frequency-throughput functional relation that is found via infinitesimal perturbation analysis (IPA). Simulation results show that the proposed algorithm can precisely regulate the throughput of out-of-order processors under a wide range of workload variations.

Next, the dissertation proposes a solution to the problem of chip power control, which is concerned with maximizing the performance of manycore chips while ensuring the power envelope stays below the chip power budget. Solving the chip power control problem has central implications on the design of the processor cooling and power delivery systems, and maximizing the performance gains of the manycore processing paradigm. The proposed solution methodology decomposes chip power control into a

master problem, which is concerned with calculating a performance-maximizing partition of the chip power budget between cores, and *regulation* subproblems, which are concerned with tracking the fractions of the power budget (power setpoints) assigned to each core. The regulation subproblems are solved using the core-power regulators described earlier. The master problem is solved using an iterative constrained-optimization scheme based on the method of gradient projection that calculates the power setpoints using the sample power-throughput derivative calculated at each core. The power setpoints calculated by the master algorithm satisfy the chip power budget constraint, as well the the local core-power constraints arising from the finite DVFS settings. Simulation results, performed using a detailed multicore processor simulator and industry-standard benchmarks, show that the proposed solution controls the power envelope more precisely and yields higher performance than state of the art.

Finally, to optimize the energy consumption of cache memories, we propose an iterative optimization algorithm based on the method of gradient descent. The proposed algorithm is an adaptive version of the cache decay technique , which switches off cache lines predicted to be unused to save their leakage energy, but may result in energy overheads in case of mispredictions. The proposed algorithm adaptively sets the cache-decay parameters to balance the tradeoff between cache leakage-energy savings and the energy overheads of induced misses, thereby optimizing cache energy under variable access patterns.

7.2 Sources and Impact of Derivative-Estimation Error

Perfect estimation of power and performance derivatives may not be attainable in practice, which merits discussing the sources of estimation error and their impact on the control and optimization algorithms proposed in this dissertation. Power-derivative error is strongly dependent on the accuracy of runtime core-power estimation, which is carried out via direct measurement or activity-based prediction.

The inclusion of runtime core-power estimation into processor platforms (e.g. Intel Sandybridge [52]) signals that their accuracy have reached an acceptable level from a practical standpoint. Moreover, as shown by the formal analysis and simulation results in Chapter 3, the proposed core-power regulation algorithm is highly robust, achieving rapid tracking under high levels of power-derivative estimation errors.

Throughput-derivative estimation is influenced by several factors pertaining to the workload and the underlying microarchitecture. The analysis carried out in Chapter 4 showed a correlation between the relative error and the rate of memory operations in the workload. Estimation errors may also arise due to architectural features or events that were not explicitly included in the out-of-order queuing model discussed in Chapter 4; e.g. speculation, or stalls due to resource hazards. As the analysis in Chapter 4 demonstrates, the proposed throughput regulation algorithm can maintain the desirable properties of rapid tracking under these error conditions.

The impact of derivative-estimation error maybe more pronounced on the chip power control algorithm discussed in Chapter 5. The performance-maximizing partition of the chip power budget is calculated using the ratio of the core throughput to power derivatives, which may experience varying levels of estimation error that affect the quality of the solution. The extensive simulation results presented in Chapter 5 demonstrate that the proposed chip power control algorithm delivers higher performance compared to the state of the art under these error conditions. Moreover, the trend of adopting simpler core microarchitectures [64, 57] is expected to alleviate the error levels especially in the throughput-derivative estimation.

Finally, the behavior of programs alternates between stable and variable-length regions called *phases* [65]. Phase-detection techniques can inform the choice of the sampling interval to ensure an adequate number of samples per phase, and therefore balance the precision of the derivative estimate and the responsiveness of the optimization and control algorithms.

APPENDIX A

POWER TRACKING PROOFS

This appendix provides the proofs for the core-power tracking results outlined in Chapter 3, and discusses further results pertaining to the tracking robustness under derivative-estimation errors. For reading convenience, part of the exposition already discussed in Chapter 3 is reintroduced in the sequel. Consider the discrete-time scalar system shown in Fig. 44, where the plant is modeled as a memoryless, time-varying nonlinear system of the form $P = g_n(\phi)$; n denotes (discrete) time and $g_n : R \to R$ is the function defining the system at time n.

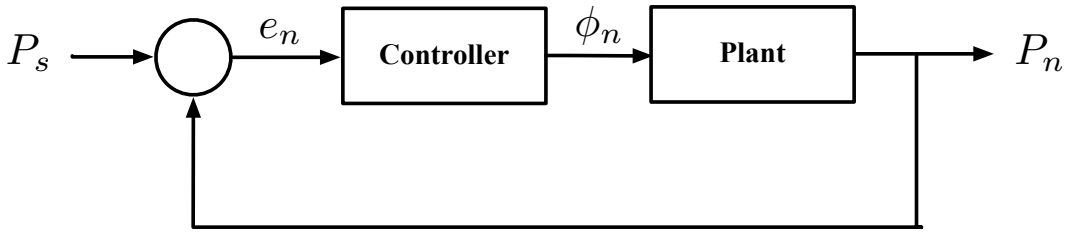


Figure 44: Power control system

Suppose that the functions g_n, $n = 1, 2 \ldots$, have a common domain, $I := [\phi_{min}, \phi_{max}]$, where the following assumption is in force.

Assumption 1. *Each one of the functions g_n is continuously differentiable, convex, and monotone-increasing throughout I. Furthermore, there exist constants $\gamma_1 > 0$ and $\gamma_2 < \infty$ such that, for every $n = 1, 2, \ldots$, $g_n'(\phi_{min}) \geq \gamma_1$, and $g_n'(\phi_{max}) \leq \gamma_2$ ('prime' denotes derivative with respect to ϕ).*

The proposed control law is a recursive computation defined by Equations (1) - (4). If the plant is time invariant, the control is effectively Newton's method for finding a zero of the equation $e = P_s - g(\phi) = 0$. In this case, we have the following

well-known result:

Proposition 2. *Suppose that the plant is time invariant. Then there exists a positive constant $\beta < 1$ such that, for every $n = 1, 2, \ldots,$*

1. *If $e_{n-1} \geq 0$ then $e_n \leq 0$.*

2. *If $e_{n-1} \leq 0$ then*

$$\beta e_{n-1} \;\leq\; e_n \;\leq\; 0. \tag{71}$$

This result is a special case of Proposition 3, below, concerning time-varying systems. As a corollary, it follows that the output tracks the reference input, since $\lim_{n\to\infty} e_n = 0$ and hence $\lim_{n\to\infty} P_n = P_s$. Moreover, this convergence is exponential in the sense that $|e_n| \leq A\beta^n$ for some $A > 0$ and $\beta \in (0, 1)$.

Consider now the time-varying case, where the closed-loop system is defined via Equations (1) - (4). The error term e_n satisfies the following inequalities.

Proposition 3. *There exists a positive constant $\beta < 1$ such that, for every $n = 1, 2, \ldots,$*

1. *If $e_{n-1} \geq 0$, then*

$$e_n \;\leq\; g_{n-1}(\phi_{n-1}) - g_n(\phi_{n-1}). \tag{72}$$

2. *If $e_{n-1} \leq 0$, then*

$$\begin{aligned} &\beta e_{n-1} + \big(g_{n-1}(\phi_{n-1}) - g_n(\phi_{n-1})\big) \\ &\leq\; e_n \;\leq\; g_{n-1}(\phi_{n-1}) - g_n(\phi_{n-1}). \end{aligned} \tag{73}$$

Proof. Consider a differentiable convex function $g : R \to R$. By the definition of convexity, for every $x \in R$ and $\Delta x \geq 0$, the following inequalities are in force:

$$g^{'}(x)\Delta x \;\leq\; g(x+\Delta x)-g(x) \;\leq\; g^{'}(x+\Delta x)\Delta x. \tag{74}$$

By (4) and Assumption 1, $K_n > 0$ for every $n = 1, 2, \ldots$.

Consider first part (1) of the proposition. Suppose that $e_{n-1} \geq 0$. By the left inequality of (74), $g_n(\phi_{n-1} + K_n e_{n-1}) \geq g_n(\phi_{n-1}) + g_n^{'}(\phi_{n-1}) K_n e_{n-1}$, and hence, and by (3), (1), and (4),

$$\begin{aligned} e_n \;&\leq\; P_s - g_n(\phi_{n-1}) - g_n^{'}(\phi_{n-1}) K_n e_{n-1} \\ &=\; P_s - g_n(\phi_{n-1}) - e_{n-1}. \end{aligned} \tag{75}$$

Subtracting and adding $g_{n-1}(\phi_{n-1})$ to the RHS of (75), and using (3) with $n-1$, Equation (72) follows.

Next, consider part (2) of the proposition. Suppose that $e_{n-1} \leq 0$. By (1) - (2),

$$e_n \;=\; P_s - P_n \;=\; P_s - g_n(\phi_n) \;=\; P_s - g_n(\phi_{n-1} + K_n e_{n-1}). \tag{76}$$

We next apply Equation (74) with $x = \phi_{n-1} + K_n e_{n-1}$ and $x + \Delta x = \phi_{n-1}$; note that $\Delta x := -K_n e_{n-1} \geq 0$. The left inequality of (74) implies, together with (1), that

$$g_n(\phi_{n-1} + K_n e_{n-1}) \;\leq\; g_n(\phi_{n-1}) + g_n^{'}(\phi_n) K_n e_{n-1}. \tag{77}$$

Consequently, and by (3) and (1),

$$e_n \;\geq\; P_s - g_n(\phi_{n-1}) - g_n^{'}(\phi_n) K_n e_{n-1}. \tag{78}$$

Subtracting and adding $g_{n-1}(\phi_{n-1})$ to the above equation we obtain that

$$
\begin{aligned}
e_n \;&\geq\; P_s - g_{n-1}(\phi_{n-1}) + g_{n-1}(\phi_{n-1}) - g_n(\phi_{n-1}) \\
&\qquad\qquad -g_n^{'}(\phi_n)K_n e_{n-1} \\
&= \Big(1 - \frac{g_n^{'}(\phi_n)}{g_n^{'}(\phi_{n-1})}\Big)e_{n-1} + g_{n-1}(\phi_{n-1}) - g_n(\phi_{n-1}), \qquad (79)
\end{aligned}
$$

where the last equality follows from (3) and (4). By (1), $\phi_{n-1} \geq \phi_n$ and hence $g_n^{'}(\phi_n) \leq g_n^{'}(\phi_{n-1})$, namely $\frac{g_n^{'}(\phi_n)}{g_n^{'}(\phi_{n-1})} \leq 1$. By Assumption 1 there exists $\alpha \in (0,1)$, independent of n, such that $\frac{g_n^{'}(\phi_n)}{g_n^{'}(\phi_{n-1})} \geq \alpha$. Defining $\beta = 1 - \alpha$, the left inequality of (73) follows from (79).

The right inequality of (73) is proved in a similar way to (72). By the right inequality of (74), we have that

$$
\begin{aligned}
e_n \;&=\; P_s - P_n \;=\; P_s - g_n(\phi_n) \\
&=\; P_s - g_n(\phi_{n-1} + K_n e_{n-1}) \\
&\leq\; P_s - g_n(\phi_{n-1}) - g_n^{'}(\phi_{n-1})K_n e_{n-1} \\
&=\; P_s - g_{n-1}(\phi_{n-1}) + g_{n-1}(\phi_{n-1}) - g_n(\phi_{n-1}) - e_{n-1} \\
&=\; g_{n-1}(\phi_{n-1}) - g_n(\phi_{n-1}), \qquad (80)
\end{aligned}
$$

thereby establishing the right inequality of (73) and completing the proof. □

Proposition 2 implies that P_n converges exponentially fast toward a band (tolerance) around the target level P_s, and the width of the band depends on how fast the plant-equation (2) varies. To see this, suppose that there exists $\varepsilon > 0$ such that for every $n = 1, 2, \ldots$, $|g_n(\phi_{n-1}) - g_n(\phi_n)| < \varepsilon$. Then Proposition 2 implies that, for every $n \geq 2$,

$$
-\frac{1}{1-\beta}\varepsilon \;\leq\; \liminf_{n\to\infty} e_n \;\leq\; \limsup_{n\to\infty} e_n \;\leq\; \varepsilon. \qquad (81)
$$

Certainly no perfect tracking can be obtained when the system is time varying, but Equation (81) shows that when the system varies slowly, namely ε is small, a narrow band can be approached. In particular, when $\varepsilon = 0$, $\lim_{n\to\infty} P_n = P_s$.

Now suppose that the controller's gain K_n is not computed exactly, but rather is estimated by a quantity $\bar{K}_n > 0$. In this case the control equation (1) is modified to the following equation,

$$\phi_n \;=\; \phi_{n-1} + \bar{K}_n e_{n-1}. \tag{82}$$

The following result is an extension of Proposition 3 and its proof is similar and hence omitted.

Proposition 4. *Let $\alpha \in (0,1]$ be as in the proof of Proposition 2, namely, for every $\phi_1 \in I$ and $\phi_2 \in I$ such that $\phi_2 \geq \phi_1$, and for every $n = 1, \ldots, \frac{g_n'(\phi_1)}{g_n'(\phi_2)} \geq \alpha$; by Assumption 1 such α exists. For every $n = 1, 2, \ldots$,*

1. *If $e_{n-1} \geq 0$, then*

$$e_n \;\leq\; \Big(1 - \frac{\bar{K}_n}{K_n}\Big)e_{n-1} + \big(g_{n-1}(\phi_{n-1}) - g_n(\phi_{n-1})\big). \tag{83}$$

2. *If $e_{n-1} \leq 0$, then*

$$\begin{aligned}\Big(1 - \alpha\frac{\bar{K}_n}{K_n}\Big)e_{n-1} + \big(g_{n-1}(\phi_{n-1}) - g_n(\phi_{n-1})\big) \\ \leq\; e_n \;\leq\; \Big(1 - \frac{\bar{K}_n}{K_n}\Big)e_{n-1} \\ +\big(g_{n-1}(\phi_{n-1}) - g_n(\phi_{n-1})\big).\end{aligned} \tag{84}$$

Observe that if $\bar{K}_n = K_n$ then Equations (83) and (84) reduce to (72) and (73) (with $\beta = 1 - \alpha$), respectively.

Suppose that there exists numbers μ and η such that $0 < \mu < \eta < 2$, and suppose that $\mu \leq \frac{\bar{K}_n}{K_n} \leq \eta$ for all $n = 1, 2, \ldots$. Suppose also that there exists $\varepsilon > 0$ such that,

for every $n = 2, \ldots, |g_n(\phi_{n-1}) - g_n(\phi_n)| \leq \varepsilon$. Then simple algebra yields the following inequalities,

$$-\frac{1}{\alpha\mu}\varepsilon \;\leq\; \liminf_{n\to\infty} e_n \;\leq\; \limsup_{n\to\infty} e_n \;\leq\; \frac{1}{\mu}\varepsilon. \tag{85}$$

Note that this equation reduces to (81) when the computation of K_n is exact, namely $\bar{K}_n = K_n$. Also, (85) is an extension of one of the convergence results in [46] where the system is linear and $\varepsilon = 0$.

APPENDIX B

FREQUENCY PARTITIONING PROOF

The core frequency allocation; given in equation (59), can lead to frequency values that lie outside the permissible range $[\phi_{min}, \phi_{max}]$. To illustrate this possibility, observe that for an N core processor, the range of f is:

$$N\phi_{min} \leq f \leq N\phi_{max}. \tag{86}$$

From equation (59), we get:

$$f = \phi_i \frac{\sum_{j=1}^{N} \alpha_j}{\alpha_i}, \tag{87}$$

Plugging the above in equation (86) yields:

$$\frac{N\alpha_i}{\sum_{j=1}^{N} \alpha_j} \phi_{min} \leq \phi_i \leq \frac{N\alpha_i}{\sum_{j=1}^{N} \alpha_j} \phi_{max}. \tag{88}$$

Hence, the range of core frequency values ϕ_i calculated by the baseline algorithm is given by the inequality above. However, for correct core frequency assignments, ϕ_i must lie within the range:

$$\phi_{min} \leq \phi_i \leq \phi_{max}. \tag{89}$$

GIven that the α_i's are time varying and their values cannot be ascertained *apriori*, there is no guarantee that core frequency values calculated by the baseline algorithm will satisfy the practical range in Equation (89).

APPENDIX C

μ CALCULATION ALGORITHM

This appendix presents an exact algorithm adapted from [60] to find the root of the piecewise linear equation (52) necessary for the calculation of the direction vectors within the master algorithm. The computation of μ starts by assuming that all direction-vector constraints are inactive, i.e. $\mu \leftarrow \sum_{i=1}^{N} \frac{\partial \mathcal{G}(S)}{\partial s_i}$. The direction vectors are then calculated as $X[i] = \frac{\partial \mathcal{G}(S)}{\partial s_i} - \mu$, and are inspected for *violations*, which occur if the the calculated direction vector yields a positive value of $f_j(X) : j \in [L+1, L+U]$. The direction-vector element with largest violation magnitude as well as its utility derivative are forced to zero, and the process is repeated until there are no violations. The inputs to the algorithm are the the vector of utility derivatives $\zeta'(S)$, and the a labeling vector A that indicates the constraints on the sign of individual direction vectors. Any element $a_i \in A$ can take a value (label) from the set $\{-, +.\pm\}$, where the values respectively denote strict non-positiveness, non-negativeness, or no sign restrictions. Details of the algorithm are given next.

Algorithm 1 Direction Vector Calculation Algorithm

function CalculateDirectionVector(A, $\zeta'(S)$)
  $X \leftarrow \Phi$ ▷ active direction vectors
  $\bar{X} \leftarrow \Phi$ ▷ direction vectors set to zero
  $v \leftarrow True$ ▷ violator flag
  $\hat{N} \leftarrow N$

  while $v = True$ **do**
    $\mu \leftarrow \frac{1}{\hat{N}} \sum_{i=1}^{\hat{N}} \zeta'(s_i)$

    for $i \leftarrow 1, \hat{N}$ **do**
      $X[i] = \zeta'(s_i) - \mu$

    $violator \leftarrow 0$

    for $i \leftarrow 1, \hat{N}$ **do** ▷ find the largest violator
      if $(X[i] > 0) \wedge (A[i] =$ **-**$) \vee (X[i] < 0) \wedge (A[i] = +)$ **then**
        if $\left|X[i]\right| > violator$ **then**
          $violator \leftarrow |X[i]|$
          $j = i$
    if $violator > 0$ **then**
      $\hat{N} = \hat{N} - 1$
      $\zeta'(S) \leftarrow \zeta'(S) \setminus \zeta'_j(s_j)$
      $X \leftarrow X \backslash X[j]$
      $\bar{X} \leftarrow \bar{X} \cup 0$
      $v \leftarrow True$
    else
      return $X \cup \bar{X}$

REFERENCES

[1] G. Moore, "Cramming more components onto integrated circuits," *Proceedings of the IEEE*, vol. 86, no. 1, pp. 82–85, 1998.

[2] R. Dennard, F. Gaensslen, H.-N. Yu, V. LEO RIDEOVT, E. Bassous, and A. R. Leblanc, "Design of ion-implanted mosfet's with very small physical dimensions," *Solid-State Circuits Society Newsletter, IEEE*, vol. 12, no. 1, pp. 38–50, 2007.

[3] M. Pedram and S. Nazarian, "Thermal modeling, analysis, and management in vlsi circuits: Principles and methods," *Proceedings of the IEEE*, vol. 94, no. 8, pp. 1487 –1501, Aug. 2006.

[4] P. Bose, "Power wall," in *Encyclopedia of Parallel Computing*, D. Padua, Ed. Springer US, 2011, pp. 1593–1608. [Online]. Available: http://dx.doi.org/10.1007/978-0-387-09766-4_499

[5] G. Hutcheson, "The economic implications of moore's law," in *High Dielectric Constant Materials*. Springer, 2005, pp. 1–30.

[6] D. Patterson and J. Hennessy, *Computer Organization and Design, Revised Fourth Edition: The Hardware/Software Interface*, ser. Morgan Kaufmann Series in Computer Graphics. Elsevier Science, 2011.

[7] H. Esmaeilzadeh, E. Blem, R. St. Amant, K. Sankaralingam, and D. Burger, "Dark silicon and the end of multicore scaling," in *Proceedings of the 38th annual international symposium on Computer architecture (ISCA '11)*. New York, NY, USA: ACM, 2011, pp. 365–376.

[8] Z. Lu, J. Hein, M. Humphrey, M. Stan, J. Lach, and K. Skadron, "Control-theoretic dynamic frequency and voltage scaling for multimedia workloads," in *Proceedings of the 2002 international conference on Compilers, architecture, and synthesis for embedded systems*, ser. CASES '02. New York, NY, USA: ACM, 2002, pp. 156–163.

[9] L. Barroso and U. Holzle, "The case for energy-proportional computing," *Computer*, vol. 40, no. 12, pp. 33 –37, Dec. 2007.

[10] C. Lefurgy, X. Wang, and M. Ware, "Server-level power control," in *Autonomic Computing, 2007. ICAC '07. Fourth International Conference on*, June 2007, p. 4.

[11] R. Raghavendra, P. Ranganathan, V. Talwar, Z. Wang, and X. Zhu, "No "power" struggles: coordinated multi-level power management for the data

center," in *Proceedings of the 13th international conference on Architectural support for programming languages and operating systems*, ser. ASPLOS XIII. New York, NY, USA: ACM, 2008, pp. 48–59. [Online]. Available: http://doi.acm.org/10.1145/1346281.1346289

[12] X. Wang, M. Chen, and X. Fu, "Mimo power control for high-density servers in an enclosure," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 21, no. 10, pp. 1412 –1426, Oct. 2010.

[13] X. Wang, M. Chen, C. Lefurgy, and T. Keller, "Ship: A scalable hierarchical power control architecture for large-scale data centers," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 23, no. 1, pp. 168 –176, Jan. 2012.

[14] K. Ma, X. Li, M. Chen, and X. Wang, "Scalable power control for many-core architectures running multi-threaded applications," in *Proceedings of the 38th annual international symposium on Computer architecture*, ser. ISCA '11. New York, NY, USA: ACM, 2011, pp. 449–460.

[15] D. Brooks, R. Dick, R. Joseph, and L. Shang, "Power, thermal, and reliability modeling in nanometer-scale microprocessors," *Micro, IEEE*, vol. 27, no. 3, pp. 49–62, 2007.

[16] R. Mahajan, C. pin Chiu, and G. Chrysler, "Cooling a microprocessor chip," *Proceedings of the IEEE*, vol. 94, no. 8, pp. 1476–1486, 2006.

[17] C. Isci, A. Buyuktosunoglu, C.-Y. Cher, P. Bose, and M. Martonosi, "An analysis of efficient multi-core global power management policies: Maximizing performance for a given power budget," in *Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO 39. Washington, DC, USA: IEEE Computer Society, 2006, pp. 347–358.

[18] M. Horowitz, T. Indermaur, and R. Gonzalez, "Low-power digital design," in *Low Power Electronics, 1994. Digest of Technical Papers., IEEE Symposium*, 1994, pp. 8–11.

[19] T. Burd and R. Brodersen, "Energy efficient cmos microprocessor design," in *System Sciences, 1995. Proceedings of the Twenty-Eighth Hawaii International Conference on*, vol. 1, 1995, pp. 288–297 vol.1.

[20] T. Burd, T. Pering, A. Stratakos, and R. Brodersen, "A dynamic voltage scaled microprocessor system," in *Solid-State Circuits Conference, 2000. Digest of Technical Papers. ISSCC. 2000 IEEE International*, 2000.

[21] J. A. Winter, D. H. Albonesi, and C. A. Shoemaker, "Scalable thread scheduling and global power management for heterogeneous many-core architectures," in *Proceedings of the 19th international conference on Parallel architectures and compilation techniques*, ser. PACT '10. New York, NY, USA: ACM, 2010, pp. 29–40.

[22] A. Mishra, S. Srikantaiah, M. Kandemir, and C. Das, "Cpm in cmps: Coordinated power management in chip-multiprocessors," in *High Performance Computing, Networking, Storage and Analysis (SC), 2010 International Conference for*, Nov. 2010, pp. 1 –12.

[23] A. K. Mishra, S. Srikantaiah, M. Kandemir, and C. R. Das, "Coordinated power management of voltage islands in cmps," *SIGMETRICS Perform. Eval. Rev.*, vol. 38, no. 1, pp. 359–360, Jun. 2010. [Online]. Available: http://doi.acm.org.www.library.gatech.edu:2048/10.1145/1811099.1811086

[24] Y. Wang, K. Ma, and X. Wang, "Temperature-constrained power control for chip multiprocessors with online model estimation," *ISCA '09: Proceedings of the 36th annual international symposium on Computer architecture*, June 2009.

[25] N. Almoosa, W. Song, Y. Wardi, and S. Yalamanchili, "A power capping controller for multicore processors," in *American Control Conference (ACC), 2012*, 2012, pp. 4709–4714.

[26] N. Almoosa, W. Song, S. Yalamanchili, and Y. Wardi, "Throughput regulation in multicore processors via ipa," in *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*, 2012, pp. 7267–7272.

[27] S. Kaxiras, Z. Hu, and M. Martonosi, "Cache decay: exploiting generational behavior to reduce cache leakage power," in *Computer Architecture, 2001. Proceedings. 28th Annual International Symposium on*, 2001, pp. 240–251.

[28] Q. Wu, P. Juang, M. Martonosi, and D. Clark, "Formal online methods for voltage/frequency control in multiple clock domain microprocessors," *ASPLOS-XI: Proceedings of the 11th international conference on Architectural support for programming languages and operating systems*, Dec. 2004.

[29] P. Juang, Q. Wu, L.-S. Peh, M. Martonosi, and D. W. Clark, "Coordinated, distributed, formal energy management of chip multiprocessors," in *Proceedings of the 2005 international symposium on Low power electronics and design*, ser. ISLPED '05. New York, NY, USA: ACM, 2005, pp. 127–130.

[30] Y. Zhu and F. Mueller, "Exploiting synchronous and asynchronous dvs for feedback edf scheduling on an embedded platform," *ACM Trans. Embed. Comput. Syst.*, vol. 7, pp. 3:1–3:26, Dec. 2007.

[31] J. Suh and M. Dubois, "Dynamic mips rate stabilization in out-of-order processors," *SIGARCH Comput. Archit. News*, vol. 37, pp. 46–56, June 2009.

[32] R. Ayoub, U. Ogras, E. Gorbatov, Y. Jin, T. Kam, P. Diefenbaugh, and T. Rosing, "Os-level power minimization under tight performance constraints in general purpose systems," in *Low Power Electronics and Design (ISLPED) 2011 International Symposium on*, 2011, pp. 321–326.

[33] S. Herbert and D. Marculescu, "Analysis of dynamic voltage/frequency scaling in chip-multiprocessors," in *Low Power Electronics and Design (ISLPED), 2007 ACM/IEEE International Symposium on*, Aug. 2007, pp. 38 –43.

[34] R. Teodorescu and J. Torrellas, "Variation-aware application scheduling and power management for chip multiprocessors," *ACM SIGARCH Computer Architecture News*, vol. 36, no. 3, pp. 363–374, 2008.

[35] J. Sartori and R. Kumar, "Distributed peak power management for many-core architectures," in *Design, Automation Test in Europe Conference Exhibition, 2009. DATE '09.*, 2009, pp. 1556–1559.

[36] X. Wang, K. Ma, and Y. Wang, "Adaptive power control with online model estimation for chip multiprocessors," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 22, no. 10, pp. 1681–1696, 2011.

[37] R. McGowen, C. Poirier, C. Bostak, J. Ignowski, M. Millican, W. Parks, and S. Naffziger, "Power and temperature control on a 90-nm itanium family processor," *IEEE JSSC*, vol. 41, no. 1, pp. 229 – 237, Jan. 2006.

[38] H. Zhou, M. C. Toburen, E. Rotenberg, and T. M. Conte, "Adaptive mode control: A static-power-efficient cache design," *Trans. on Embedded Computing Sys.*, vol. 2, no. 3, pp. 347–372, 2003.

[39] S. Velusamy, K. Sankaranarayanan, D. Parikh, T. Abdelzaher, , and K. Skadron, "Adaptive cache decay using formal feedback control," in *In Proceedings of the 2002 Workshop on Memory Performance Issues*, 2002.

[40] C. Meenderinck and B. Juurlink, "(when) will cmps hit the power wall?" in *Euro-Par 2008 Workshops - Parallel Processing*, ser. Lecture Notes in Computer Science, E. César, M. Alexander, A. Streit, J. Träff, C. Cérin, A. Knüpfer, D. Kranzlmüller, and S. Jha, Eds. Springer Berlin Heidelberg, 2009, vol. 5415, pp. 184–193. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-00955-6_23

[41] R. Kumar, D. Tullsen, N. Jouppi, and P. Ranganathan, "Heterogeneous chip multiprocessors," *Computer*, vol. 38, no. 11, pp. 32 – 38, Nov. 2005.

[42] Q. Deng, D. Meisner, L. Ramos, T. F. Wenisch, and R. Bianchini, "Memscale: active low-power modes for main memory," in *Proceedings of the sixteenth international conference on Architectural support for programming languages and operating systems*, ser. ASPLOS XVI. New York, NY, USA: ACM, 2011, pp. 225–238. [Online]. Available: http://doi.acm.org/10.1145/1950365.1950392

[43] S. Eyerman and L. Eeckhout, "System-level performance metrics for multiprogram workloads," *Micro, IEEE*, vol. 28, no. 3, pp. 42–53, 2008.

[44] K. Luo, J. Gummaraju, and M. Franklin, "Balancing thoughput and fairness in smt processors," in *Performance Analysis of Systems and Software, 2001. ISPASS. 2001 IEEE International Symposium on.* IEEE, 2001, pp. 164–171.

[45] R. Bitirgen, E. Ipek, and J. F. Martinez, "Coordinated management of multiple interacting resources in chip multiprocessors: A machine learning approach," in *Proceedings of the 41st annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO 41. Washington, DC, USA: IEEE Computer Society, 2008, pp. 318–329.

[46] C. Lefurgy, X. Wang, and M. Ware, "Power capping: A prelude to power shifting," *Cluster Computing*, 2008.

[47] M. S. Floyd, S. Ghiasi, T. W. Keller, K. Rajamani, F. L. Rawson, J. C. Rubio, and M. S. Ware, "System power management support in the ibm power6 microprocessor," *IBM Journal of Research and Development*, vol. 51, no. 6, pp. 733 –746, Nov. 2007.

[48] J. M. Rabaey, *Digital Integrated Circuits: A Design Perspective.* Prentice Hall, 1995.

[49] K. Mistry, C. Allen, C. Auth, B. Beattie, D. Bergstrom, M. Bost, M. Brazier, M. Buehler, A. Cappellani, R. Chau, C.-H. Choi, G. Ding, K. Fischer, T. Ghani, R. Grover, W. Han, D. Hanken, M. Hattendorf, J. He, J. Hicks, R. Huessner, D. Ingerly, P. Jain, R. James, L. Jong, S. Joshi, C. Kenyon, K. Kuhn, K. Lee, H. Liu, J. Maiz, B. Mclntyre, P. Moon, J. Neirynck, S. Pae, C. Parker, D. Parsons, C. Prasad, L. Pipes, M. Prince, P. Ranade, T. Reynolds, J. Sandford, L. Shifren, J. Sebastian, J. Seiple, D. Simon, S. Sivakumar, P. Smith, C. Thomas, T. Troeger, P. Vandervoorn, S. Williams, and K. Zawadzki, "A 45nm logic technology with high-k metal gate transistors, strained silicon, 9 cu interconnect layers, 193nm dry patterning, and 100% pb-free packaging," in *IEEE International Electron Devices Meeting 2007 (IEDM 2007)*, Dec. 2007.

[50] R. Kuppuswamy, S. Sawant, S. Balasubramanian, P. Kaushik, N. Natarajan, and J. Gilbert, "Over one million tpcc with a 45nm 6-core xeon," in *Solid-State Circuits Conference - Digest of Technical Papers, 2009. ISSCC 2009. IEEE International*, Feb. 2009, pp. 70 –71,71a.

[51] S. Sawant, U. Desai, G. Shamanna, L. Sharma, M. Ranade, A. Agarwal, S. Dakshinamurthy, and R. Narayanan, "A 32nm westmere-ex xeon enterprise processor," in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2011 IEEE International*, Feb. 2011, pp. 74 –75.

[52] E. Rotem, A. Naveh, D. Rajwan, A. Ananthakrishnan, and E. Weissmann, "Power-management architecture of the intel microarchitecture code-named sandy bridge," *Micro, IEEE*, vol. 32, no. 2, pp. 20–27, 2012.

[53] G. Loh, S. Subramaniam, and Y. X., "Zesto: A cycle-level simulator for highly detailed microarchitecture exploration," in *ISPASS 2009.*, April 2009, pp. 53 –64.

[54] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi, "Mcpat: an integrated power, area, and timing modeling framework for multicore and manycore architectures," in *IEEE MICRO'09*, 2009, pp. 469–480.

[55] R. Wilhelm, J. Engblom, A. Ermedahl, N. Holsti, S. Thesing, D. Whalley, G. Bernat, C. Ferdinand, R. Heckmann, T. Mitra, F. Mueller, I. Puaut, P. Puschner, J. Staschulat, and P. Stenström, "The worst-case execution-time problem–overview of methods and survey of tools," *ACM Trans. Embed. Comput. Syst.*, vol. 7, pp. 36:1–36:53, May 2008.

[56] S. Eyerman, L. Eeckhout, T. Karkhanis, and J. E. Smith, "A mechanistic performance model for superscalar out-of-order processors," *ACM Trans. Comput. Syst.*, vol. 27, no. 2, pp. 3:1–3:37, May 2009.

[57] S. Borkar, "Thousand core chips: a technology perspective," in *Proceedings of the 44th annual Design Automation Conference*, ser. DAC '07. New York, NY, USA: ACM, 2007, pp. 746–749. [Online]. Available: http://doi.acm.org/10.1145/1278480.1278667

[58] M. Hill and M. Marty, "Amdahl's law in the multicore era," *Computer*, vol. 41, no. 7, pp. 33 –38, July 2008.

[59] S. Boyd and L. Vandenberghe, *Convex Optimization.* New York, NY, USA: Cambridge University Press, 2004.

[60] D. Palomar and J. Fonollosa, "Practical algorithms for a family of waterfilling solutions," *Signal Processing, IEEE Transactions on*, vol. 53, no. 2, pp. 686 – 695, feb 2005.

[61] S. Ramaswamy and S. Yalamanchili, "An utilization driven framework for energy efficient caches." in *HiPC*, ser. Lecture Notes in Computer Science, P. Sadayappan, M. Parashar, R. Badrinath, and V. K. Prasanna, Eds., vol. 5374. Springer, 2008, pp. 583–594.

[62] H. J. Kushner and G. G. Yin, *Stochastic Approximation Algorithms and Applications.* New York, NY: Springer-Verlag, 1997.

[63] M. Zhang, K. Parikh, K. Sankaranarayanan, K. Skadron, and M. R. Stan, "Hotleakage: An architectural, temperature-aware model of subthreshold and gate leakage," University of Virginia Dept. of Computer Science, Tech. Rep. CS-2003-05, March 2003.

[64] T. Morad, U. Weiser, A. Kolodnyt, M. Valero, and E. Ayguade, "Performance, power efficiency and scalability of asymmetric cluster chip multiprocessors," *Computer Architecture Letters*, vol. 5, no. 1, pp. 14 –17, June 2006.

[65] T. Sherwood, E. Perelman, G. Hamerly, S. Sair, and B. Calder, "Discovering and exploiting program phases," *Micro, IEEE*, vol. 23, no. 6, pp. 84–93, 2003.

[66] M. Cho, N. Sathe, M. Gupta, S. Kumar, S. Yalamanchilli, and S. Mukhopadhyay, "Proactive power migration to reduce maximum value and spatiotemporal non-uniformity of on-chip temperature distribution in homogeneous many-core processors," in *Semiconductor Thermal Measurement and Management Symposium, 2010. SEMI-THERM 2010. 26th Annual IEEE*, 2010, pp. 180–186.

[67] K. Ma and X. Wang, "Pgcapping: exploiting power gating for power capping and core lifetime balancing in cmps," in *Proceedings of the 21st international conference on Parallel architectures and compilation techniques*, ser. PACT '12. New York, NY, USA: ACM, 2012, pp. 13–22. [Online]. Available: http://doi.acm.org/10.1145/2370816.2370821

[68] J. Sharkey, A. Buyuktosunoglu, and P. Bose, "Evaluating design tradeoffs in on-chip power management for cmps," in *Proceedings of the 2007 international symposium on Low power electronics and design*, ser. ISLPED '07. New York, NY, USA: ACM, 2007, pp. 44–49. [Online]. Available: http://doi.acm.org/10.1145/1283780.1283791

[69] A. K. Coskun, R. Strong, D. M. Tullsen, and T. Simunic Rosing, "Evaluating the impact of job scheduling and power management on processor lifetime for chip multiprocessors," in *Proceedings of the eleventh international joint conference on Measurement and modeling of computer systems*, ser. SIGMETRICS '09. New York, NY, USA: ACM, 2009, pp. 169–180.

[70] H. Vandierendonck and T. Mens, "Techniques and tools for parallelizing software," *Software, IEEE*, vol. 29, no. 2, pp. 22 –25, April 2012.

[71] D. P. Bertsekas, *Nonlinear Programming*, 2nd ed. Nashua, New Hampshire: Athena Scientific, 1999.

[72] J. Howard, S. Dighe, S. Vangal, G. Ruhl, N. Borkar, S. Jain, V. Erraguntla, M. Konow, M. Riepen, M. Gries, G. Droege, T. Lund-Larsen, S. Steibl, S. Borkar, V. De, and R. Van Der Wijngaart, "A 48-core ia-32 processor in 45 nm cmos using on-die message-passing and dvfs for performance and power scaling," *Solid-State Circuits, IEEE Journal of*, vol. 46, no. 1, pp. 173 –183, Jan. 2011.

[73] F. Vázquez-Abad, "A course on sensitivity analysis for gradient estimation of des performance measures," in *Discrete Event Systems: Analysis and Control*, R. Boel and G. Stremersch, Eds. Boston, Massachusetts: Kluwer Academic Publishers, 2000.

[74] Y. Wardi and G. Riley, "Infinitesimal perturbation analysis in networks of stochastic flow models: General framework and case study of tandem networks with flow control," *Discrete Event Dynamic Systems*, vol. 20, pp. 275–305, 2010.

[75] Y. Ho and X. Cao, *Perturbation Analysis of Discrete Event Dynamic Systems*. Boston, Massachusetts: Kluwer Academic Publishers, 1991.

[76] C. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*. Boston, Massachusetts: Kluwer Academic Publishers, 1999.

[77] C. Cassandras, Y. Wardi, B. Melamed, G. Sun, and C. Panayiotou, "Perturbation analysis for online control and optimization of stochastic fluid models," *Automatic Control, IEEE Transactions on*, vol. 47, no. 8, pp. 1234 – 1248, Aug. 2002.

[78] Y.-S. Lin and D. Sylvester, "Runtime leakage power estimation technique for combinational circuits," in *Design Automation Conference, 2007. ASP-DAC '07. Asia and South Pacific*, Jan. 2007, pp. 660 –665.

[79] Y. Liu, R. Dick, L. Shang, and H. Yang, "Accurate temperature-dependent integrated circuit leakage power estimation is easy," in *Design, Automation Test in Europe Conference Exhibition, 2007. DATE '07*, April 2007, pp. 1 –6.

[80] C.-K. Tseng, S.-Y. Huang, C.-C. Weng, S.-C. Fang, and J.-J. Chen, "Black-box leakage power modeling for cell library and sram compiler," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2011*, March 2011, pp. 1 –6.

[81] A. Kansal, F. Zhao, J. Liu, N. Kothari, and A. A. Bhattacharya, "Virtual machine power metering and provisioning," in *Proceedings of the 1st ACM symposium on Cloud computing*, ser. SoCC '10. New York, NY, USA: ACM, 2010, pp. 39–50.

[82] M. Ware, K. Rajamani, M. Floyd, B. Brock, J. Rubio, F. Rawson, and J. Carter, "Architecting for power management: The ibm power7 approach," in *High Performance Computer Architecture (HPCA), 2010 IEEE 16th International Symposium on*, Jan. 2010, pp. 1 –11.

[83] M. Powell, A. Biswas, J. Emer, S. Mukherjee, B. Sheikh, and S. Yardi, "Camp: A technique to estimate per-structure power at run-time using a few simple parameters," in *High Performance Computer Architecture, 2009. HPCA 2009. IEEE 15th International Symposium on*, Feb. 2009, pp. 289 –300.

[84] H. Hoffmann, S. Sidiroglou, M. Carbin, S. Misailovic, A. Agarwal, and M. Rinard, "Dynamic knobs for responsive power-aware computing," in *Proceedings of the sixteenth international conference on Architectural support for programming languages and operating systems*, ser. ASPLOS '11. New York, NY, USA: ACM, 2011, pp. 199–212.

[85] R. Katz, "Tech titans building boom," *Spectrum, IEEE*, vol. 46, no. 2, pp. 40 –54, Feb. 2009.

[86] M. Ghasemazar, E. Pakbaznia, and M. Pedram, "Minimizing energy consumption of a chip multiprocessor through simultaneous core consolidation and dvfs," in *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*, June 2010, pp. 49 –52.

[87] L. Thiele, S. Chakraborty, and A. Maxiaguine, "Dvs for buffer-constrained architectures with predictable qos-energy tradeoffs," in *Hardware/Software Codesign and System Synthesis, 2005. CODES+ISSS '05. Third IEEE/ACM/IFIP International Conference on*, Sept. 2005, pp. 111 –116.

[88] M. Saravana, S. Govidan, C. Lefurgy, and A. Dholakia, "Using on-line power modeling for server power capping," *Workshop on Energy-Efficient Design (WEED 2009)*, 2009.

[89] R. Raghavendra, P. Ranganathan, V. Talwar, Z. Wang, and X. Zhu, "No "power" struggles: coordinated multi-level power management for the data center," *SIGARCH Comput. Archit. News*, vol. 36, pp. 48–59, March 2008.

[90] W. Felter, K. Rajamani, T. Keller, and C. Rusu, "A performance-conserving approach for reducing peak power consumption in server systems," in *In Proceedings of ICS*, 2005, pp. 293–302.

[91] A. Hergenhan and W. Rosenstiel, "Static timing analysis of embedded software on advanced processor architectures," in *Design, Automation and Test in Europe Conference and Exhibition 2000. Proceedings*, 2000, pp. 552 –559.

[92] Y. Tan and I. Mooney, V.J., "Timing analysis for preemptive multi-tasking real-time systems with caches," in *Design, Automation and Test in Europe Conference and Exhibition, 2004. Proceedings*, vol. 2, Feb. 2004, pp. 1034 – 1039 Vol.2.

[93] R. Gonzalez and M. Horowitz, "Energy dissipation in general purpose microprocessors," *Solid-State Circuits, IEEE Journal of*, vol. 31, no. 9, pp. 1277 –1284, Sept. 1996.

[94] O. Azizi, A. Mahesri, B. C. Lee, S. J. Patel, and M. Horowitz, "Energy-performance tradeoffs in processor architecture and circuit design: a marginal cost analysis," in *Proceedings of the 37th annual international symposium on Computer architecture*, ser. ISCA '10. New York, NY, USA: ACM, 2010, pp. 26–36.

[95] T. Austin, E. Larson, and D. Ernst, "Simplescalar: an infrastructure for computer system modeling," *Computer*, vol. 35, no. 2, pp. 59–67, Feb. 2002.

[96] E. Perelman, G. Hamerly, and B. Calder, "Picking statistically valid and early simulation points," *Parallel Architectures and Compilation Techniques, International Conference on*, 2003.